64'er

NOVEMBER 1984 ÖS 50,–/Sfr 6,– DM 6,–

Eine Markt & Technik Publikation

11|84 DAS MAGAZIN FÜR COMPUTER-FANS

C 64: Der Grafikgigant

★ ›Zeichenbretter‹ im Test: Koala Pad und Super Sketch

★ Test: Supergrafik 64 und Extended Graphik

★ Pseudo-Sprites für VC 20

★ Werkzeug für Grafikprogrammierer: viele Tips und Listings

Vergleich: Monitor kontra Fernseher

Marktübersicht Monitore

Listing des Monats

Zeichnen mit der schnellen Schildkröte

Maschinensprache

Besser als gekauft Monitor zum Abtippen

Zwei neue Kurse

★ Memory Map: alle POKEs im Detail

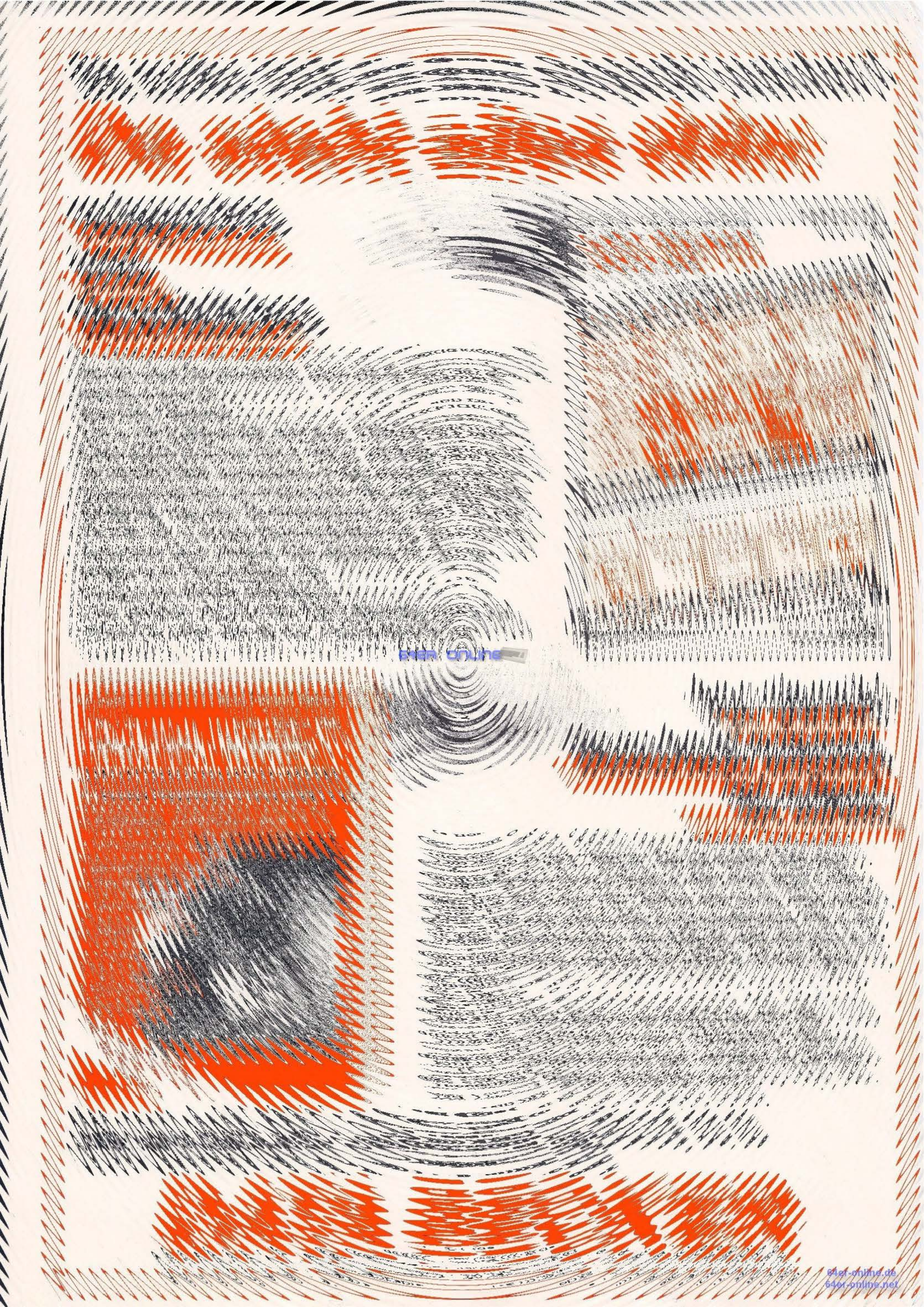
★ Einführung in Comal

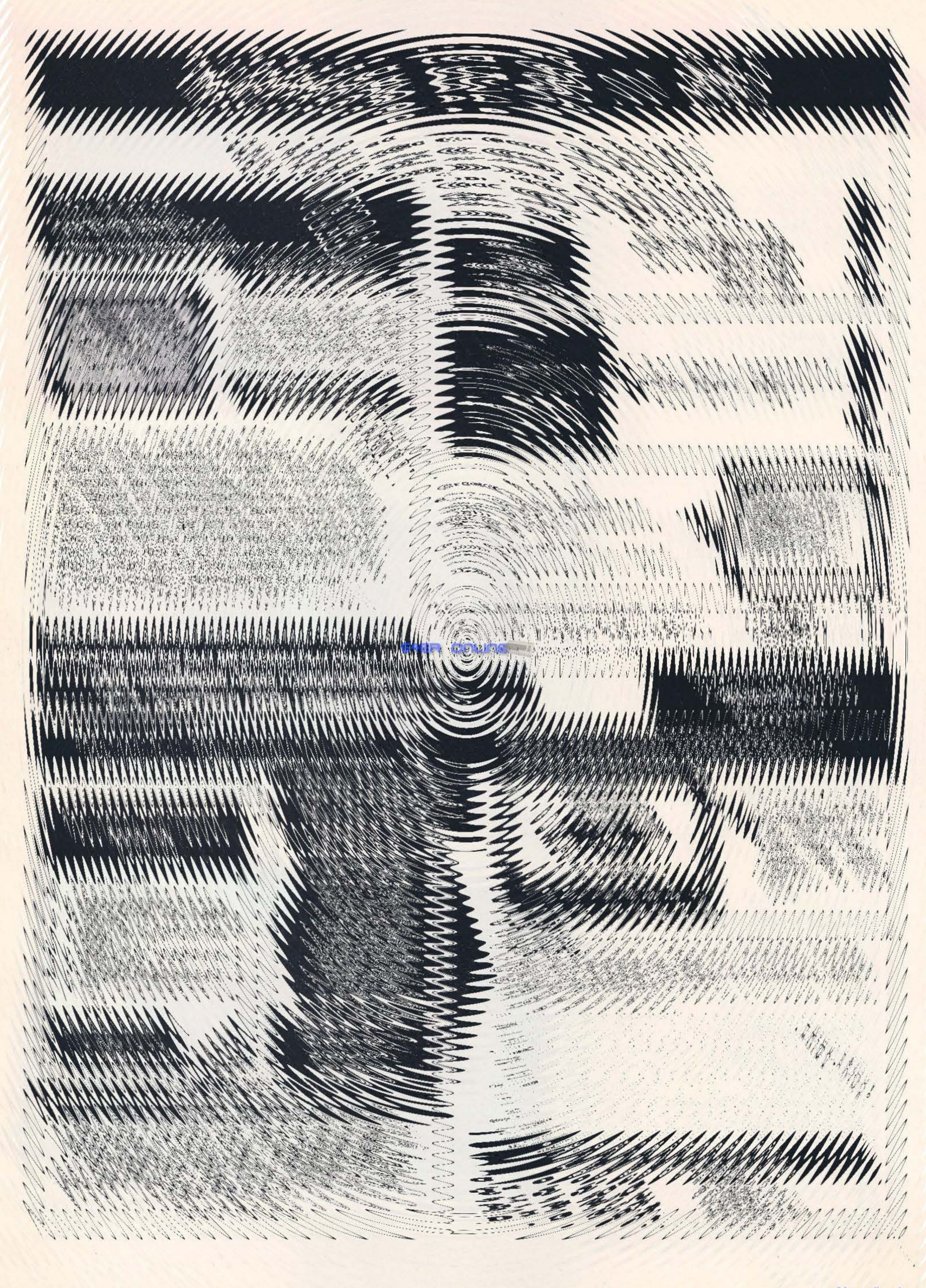
Programmierwettbewerb

Die besten Einzeiler



Schachmeister: archivieren, nachspielen, analysieren ★ Test: MSD-Floppy ★ Interrupt-Technik ★ Die Ebenen des Absturzes ★ Joystick-Abfrage ★ Tips & Tricks für den VC 20 und C 64


64er online



INHALT

Aktuell

Hardware

Hardware-Test

Software-Test

Spiele-Test

Wettbewerbe



Diese überlebensgroße »Leinwand« war nur eine der Anziehungspunkte auf der Commodore Fachausstellung 10



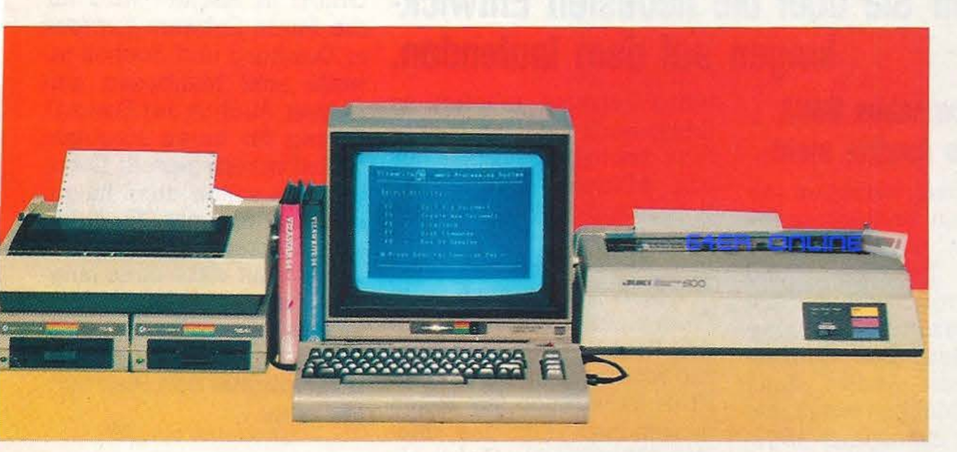
Blond oder Schwarz? Monitor oder Fernseher? Jeder nach seinem Geschmack 16



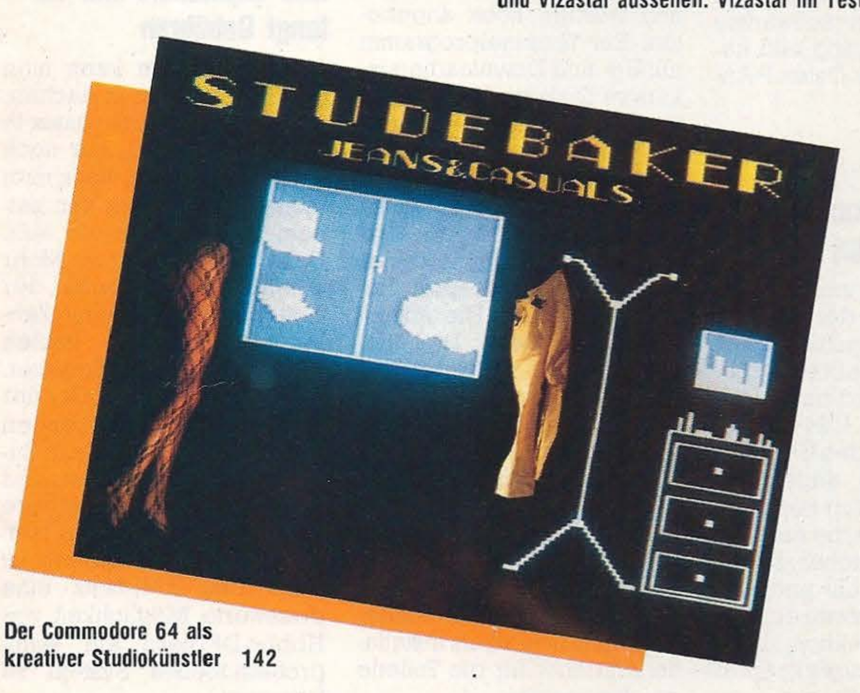
Im Test: Das MSD SD-2. Wie kompatibel ist es zur 1541? 14



Wie David gegen Goliath stehen sich das Koala Pad und das Super Sketch gegenüber. Wer wird gewinnen? 34



So könnte ein professioneller Arbeitsplatz mit Vizawrite und Vizastar aussehen. Vizastar im Test 38



Der Commodore 64 als kreativer Studiokünstler 142

Programme zum Abtippen

Anwendungen

Grafik

Tips und Tricks

Kurse

Neue Kurse

So machen's andere

Rubriken



Zuviel Programme, keine Ideen?

In Amerika stöhnen die Propheten (und vor allem die, die sich auf die Voraussagen verließen): Sie haben sich offenbar geirrt. Nach den Schätzungen von Future Computing dürften die Softwareumsätze in diesem Jahr nicht um 100 sondern »nur« um 60 Prozent wachsen. Ein anderes Marktforschungsunternehmen, Creative Strategies, schätzt bei Business Software sogar nur 40 (statt vorher über 50) Prozent — und ist bei Heimcomputer-Software noch skeptischer: Die schlechten Umsätze in diesem Sommer könnten zwar saisonbedingt sein — wahrscheinlich fehle es aber an der richtigen Art von Programmen. Ein Mitarbeiter des Verlags Prentice-Hall wurde noch deutlicher: »Es wird ein Haufen Mist angeboten« erklärte kürzlich Lynn Lumsden »Und so etwas kauft der Kunde einfach nicht mehr«. Es gebe, so meinen die Amerikaner, zu viele schlechte und unter den brauchbaren zu viele »me too«-Programme — an der xten Textverarbeitung oder der yten Dateiverwaltung bestehe kein Bedarf; sie drückten höchstens die Preise. Es fehle an guten und innovativen Produkten.

Ein kleines Beispiel dafür, daß sich der Markt anders entwickelte, als viele dachten, bietet Commodore: Mit Handbüchern wird mehr Umsatz gemacht als mit Spiel-Programmen.

Bei Heimcomputersoftware stimmt leider in vielen Fällen das Verhältnis von Nutzen und Kaufpreis nicht — zumal sich Arbeiten von Hobbyprogrammierern häufig mit dem messen können, was kommerziell vermarktet wird. Der Erfolg unseres Einzeiler-Wettbewerbs zeigt, daß es an Ideen ebensowenig mangelt, wie an der Bereitschaft, Anregungen aufzugreifen. Vielleicht sind die Anwender auf dem richtigen Weg — und nur manche Marketing-Profis auf dem Holzweg.

Michael Pauly, Chefredakteur

Aktuell

Informationen zur Datenfernübertragung

Die DFÜ steckt in Deutschland mehr oder weniger noch in den Kinderschuhen. Dennoch oder gerade deshalb halten wir Sie über die neuesten Entwicklungen auf dem laufenden.

Vertrag zwischen IMCA und Radio Austria steht

Der Vertrag zwischen IMCA, die ein professionelles System in der Nähe von Frankfurt betreiben, und dem österreichischen Staat ist vor kurzem abgeschlossen worden. Das Endprodukt, was sich dann DADAUS-Mailbox nennt, hat dann die inzwischen allgemein üblichen Fähigkeiten: Messages schicken und empfangen, ein Schwarzes Brett, Telex-Zugang und natürlich mehrere Datex-P-Anschlüsse.

Preiswerter Akustikkoppler für den C 64

Für den C 64 und den VC 20 wird derzeit der billigste 300 Baud-Akustikkoppler überhaupt angeboten. Dieses »Modul« wird unter Umgehung einer V.24-Schnittstelle direkt an den Userport des Rechners angesteckt und ist dann sofort betriebsbereit. Es bietet die für Akustikkoppler üblichen Fähigkeiten, wie CALL- und ANSWER-Modus, sowie eine zusätzliche Testfunktion, die ja nicht jeder Akustikkoppler besitzt. Das Gerät wird ohne jedes Gehäuse geliefert, es fehlen auch die »Gummimuffen«, das heißt Lautsprecher und Mikrofon hängen im Rohzustand von der Platine weg. Mit dem Gerät erhält man noch ein kleines Kommunikationsprogramm in Form eines Listings, mit dem die Sache einwandfrei betrieben werden kann. Der Preis ist wirklich fast schon sensationell: Für das »Modul« 138 oder das Ganze als Bausatz für 88 Mark. Zusätzlich werden noch angeboten: Ein Terminalprogramm mit Up- und Downloading inklusive Diskette für 25 Mark und ein kleines Interface, das dann die automatische Telefonwahl ermöglicht für 39 Mark. Das Gerät hat keine FTZ-Zulassung und darf nicht am Telefonnetz der Deutschen Bundespost betrieben werden. Die Adresse: Fotoelektronik Dipl.-Ing. Immo Drust, Landwehrstr. 5, 6100 Darmstadt.
Der Vertrieb läuft nur über den Versand, Bezahlung wie üblich per Nachnahme oder V-Scheck. Der Computertyp muß angegeben werden (C 64 oder VC 20). Noch ein Tip: brauchbare »Gummimuffen« erhält man in Sanitärfachgeschäften, die dort als Installationszubehör für die Toilette geführt werden.

RMI und Software Express arbeiten zusammen

Die RMI-Nachrichten GmbH in Aachen wird für die Firma Software-Express in Duisburg und Bocholt jeweils acht Mailboxen einrichten. Auch ist der Datex-P-Antrag für einen Hauptanschluß bereits gestellt. Diese Systeme sollen dann hauptsächlich für die Produkte des Software-Express werben. Damit wäre dann langsam die Mailbox als Werbemedium interessant geworden, da man auch als »Fremdfirma« dort Platz mieten kann.

RMI expandiert und verlangt Gebühren

Seit neuestem kann man die RMI-Mailbox in Aachen, erreichbar über Datex-P-NUA 44241040341, nur noch vernünftig nutzen, wenn man monatlich 10 Mark von seinem Konto abbuchen läßt. Dazu hat Sysop Rupert Mohr einen Verein gegründet, der sich AMDAT e.V. nennt. Zahlende Mitglieder heißen dann assoziierte Benutzer. Begründet wird die Gebühr von 10 Mark mit steigenden Kosten. So hat man dann Zugang auf das Telex-Netz und längerfristig sind mehrere Datex-P Anschlüsse geplant. Im Großen und Ganzen ist dann das RMI-Netz eine preiswerte Möglichkeit von Hobby-DFÜlern ein semiprofessionelles System zu benutzen.

Btx und Datenschutz

Btx wird (hoffentlich) bald auch für die C 64 Anwender interessant, da das Btx-Steckmodul für den C 64 allmählich auf den Markt kommen soll. Dieses Modul soll dann für zirka 150 Mark erhältlich sein. Um letztendlich Btx-Benutzer über den C 64 zu sein, ist aber noch ein Loewe oder Blaupunkt-Fernseher mit CEPT-Dekoder notwendig. Worum es bei Btx im Groben geht dürfte inzwischen fast jedem bekannt sein: Die Anbieter bieten ihre »Seiten« an, das heißt sie stellen eine bestimmte Information, mit Grafik und Farbe versteht sich, gegen Gebühren zur Verfügung.

Diese Seiten können alles mögliche beinhalten, zum Beispiel den Wetterbericht, lokale Informationen (Veranstaltungen etc.) oder auch aktuelle Nachrichten einer Zeitungsredaktion. Der Benutzer, also der, der dann mit seinem C 64 und Btx-Fernseher daheim sitzt und gebührenpflichtige Seiten abruft, muß auch dafür zahlen. Diese Gebühren werden dann praktisch auf die normale Telefonrechnung mit aufgeschlagen. Jetzt kommt — in Bezug auf den Datenschutz — der Haken an der Sache: Wer wann welche Seiten sich wie lange angeschaut hat, muß — aus Abrechnungsgründen — erfaßt werden, ganz klar. Nun ist es aber nicht nur theoretisch möglich Persönlichkeitsprofile zu erstellen, das heißt man kann aufgrund der im Btx-Rechner gespeicherten Daten ziemlich genau feststellen, welche Vorliebe und welche Abneigungen ein bestimmter Benutzer hat. Bei der Benutzung einer »Telezeitung« etwa würde auch die Zeit abrufbar sein, die sich ein Benutzer zum Beispiel im Wirtschaftsteil aufgehalten hat, und daraus könnte man auf die Verständnisfähigkeit der Person in bestimmten Gebieten schließen. Das »Bild« oder Persönlichkeitsprofil, das so von jedem Btx-Benutzer gemacht werden kann, könnte theoretisch irgendwie ausgenutzt werden. Das alles wäre überhaupt kein Problem, wenn die Abrechnungsdaten unter Aufsicht regelmäßig gelöscht würden, und genau das wird bis jetzt — obwohl vorher vereinbart — von der Bundespost abgelehnt. Klar, daß die Datenschutzbeauftragten der Länder ihre Kritik äußern. Es bleibt nur noch die Frage nach dem Grund dieser Verweigerung.

Elektronische Mitfahrerzentrale in Hamburg

Der Chaos Computer Club in Hamburg will seine elektronische Mitfahrerzentrale auf einen C 64 mit Floppy realisieren. Bei diesem System, dessen Benutzung absolut gebührenfrei sein soll, kann dann jeder, der eine Autofahrt plant, sich — natürlich nach Städten sortiert — unter Angabe seiner Telefonnummer und Adresse eintragen. Genauso können dann die Mitfahrer die Angebote abrufen und sich dann mit den Fahrern telefonisch in Verbindung setzen. Die Wahl eines C 64 bei diesem System macht das günstige Preis-Leistungs-Verhältnis des C 64 deutlich. Übrigens: Der Chaos Computer Club hofft noch auf Spenden, die die finzianziellen Aufwendungen etwas mildern sollen. Die Adresse: CCC c/o Schwarzmarkt Bundesstr. 9, 2000 Hamburg 13

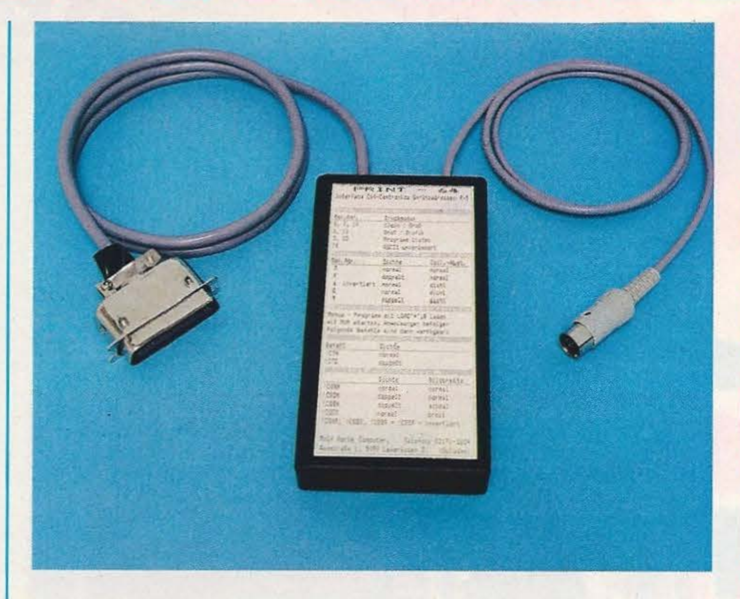
(Thomas Obermair/aa)

C 64 Interface für Centronics-Drucker

Rolf Rocke Computer bietet mit dem »Print 64« ein prozessorgesteuertes Drucker-Interface für Anspruchsvolle. Nachahmenswert ist ein auf der mitgelieferten Diskette befindlicher Druckerkurs, der den Umgang mit der Schnittstelle und den gängigsten Matrixdruckern wie Epson, Star oder Sekonic für den Anfänger anschaulich erläutert.

Außergewöhnlich ist auch die Möglichkeit der Darstellung von Farben (auf Schwarz-Weiß-Druckern) durch geeignete Wahl von zugeordneten Graustufen beim Druck von Mehrfarbgrafiken. Mit fünf unterschiedlichen Tönungen bleiben bei Mehrfarbbildern (die vom Koalapainter oder Paint Magic erstellt sein können) die Nuancen des Bildes in der Hardcopy erhalten. »Print 64« soll alle gängigen Textverarbeitungsprogramme für den C 64 unterstützen. Das heißt, auch Umlaute, Fettschrift oder Unterstreichen werden getreu wiedergegeben. Programmlistings werden mit allen Grafik- und Steuerzeichen in Grafik- oder Groß/Kleinschrift-Modus ausgeben. Der Ausdruck des Bildschirms ist in verschiedenen Varianten möglich. Der Textbildschirm kann in einfacher oder doppelter Dichte ausgegeben werden. Bilder in hochauflösender Grafik benötigen für die Übertragung sowohl in normaler als auch in doppelter Dichte nur zirka 70 Sekunden (FX 80). Durch die Wahl der Bildbreite über das gesamte A4-Format ergibt sich ein Verhältnis von Höhe zu Breite die auch die Darstellung von Kreisen orginalgetreu zuläßt. Die Sekundäradressen 3, 4, 6, 8 und 9 wurden für selbst zu erstellende Hardcopy-Routinen reserviert. Das Centronics-Interface mit der Diskette wird voraussichtlich 300 Mark kosten. (aa)

Info: Rolf Rocke Computer, Auestraße 1, 5090 Leverkusen 3, Tel.: 02171-2624



Print 64 ist ein leistungsfähiges Interface für fast alle Centronics-Drucker

Deutscher Zeichensatz für 1526 und MPS 802

Das Ingenieurbüro Hollmann in Hamburg bietet für die Commodore-Drucker 1526 und MPS 802 deutsche Zeichensätze im EPROM an. Die Zeichensätze sind auf mehrere zum C 64 erhältliche professionelle Textprogramme abgestimmt.

Zur Zeit sind deutsche Zeichensätze für SM-Text und Vizawrite verfügbar. In Vorbereitung sind Zeichensätze für Vizawrite-Benutzer in der Schweiz sowie C 64-DIN-Zeichensatz und andere Spezialzeichensätze. Die Zeichensätze können direkt im Drucker betrieben werden. Allerdings ist erst durch die Verwendung einer Umschaltplatine (Multifont, vom selben Hersteller) volle Kompatibilität gewährleistet (Textverarbeitung und Basic/Grafik-Listings). Auf der Einbauplatine lassen sich bis zu vier Zeichensätze/Betriebssysteme (im EPROM) unterbringen, die von zwei außenliegenden Schaltern ausgewählt werden. Die Zeichensätze kosten jeweils 79 Mark, die Umschaltplatine 90 Mark.

(aa)

Info: Dipl. Ing. Alfred Hollmann, Kleinfeld 36, 2104 Hamburg 92, Tel.: 040-7017483 oder 040-8501556

Commodore Fachausstellung

Commodore ist wohl der einzige Computer-Hersteller der in Deutschland eine eigene Fachmesse veranstaltet. Auf der diesjährigen vierten internationalen Commodore Fachaustellung (CFA) in Frankfurt, gab es neben Altbewährtem einige Überraschungen. Software für den Plus/4, grafisch gestaltete Diskettenhüllen, Video-Digitizer, ein Joystick ohne Boden, Musikprogramme und Keyboards, Lernsoftware oder eine schnellere 1541 sind nur einige Beispiele. Doch eins war besonders klar zu erkennen: Der C 64 dominierte an allen Ecken und Enden.



Leuchtdiodenwand als Bildschirm

Weniger die Software, sondern mehr die Datenträger erregten großes Interesse auf der CFA in Frankfurt. Bunte Disketten waren »in«, in jeder erdenklichen Farbe oder Farbkombination. Da diese Disketten auch nicht sehr viel teurer waren als normale Disketten, fanden sie natürlich reißenden Absatz. Besonders gefragt waren Memron-Disketten, die es mit allen möglichen abgebildeten Motiven gab: Weintrauben, golden mit schwarzen Streifen, bunte Schrift »Memo«, und so weiter. Einen besonderen Leckerbissen für Händler und private Großabnehemer bietet Lucius-Computer-Programme: Bei einer Abnahme von über 1000 Stück kann man sich sein eigenes Motiv auf die Diskettenhüllen drucken lassen.

Absteiger und Renner

Der VC 20 scheint nun endgültig auf dem absteigenden Ast zu sein. Ein Vertreter einer englischen Softwarefirma wunderte sich, warum in Deutschland überhaupt noch Software für den VC 20 angeboten wird. Selbst Commodore verzichtete darauf, irgend etwas zu zeigen, das mit dem VC 20 zu tun gehabt hätte — kein Wunder, denn der C 16 soll ja schließlich als Nachfolger für den VC 20 dessen Rolle übernehmen. Der Plus/4, jetzt mit vier fest eingebauten Programmen, war ebenfalls von Menschentrauben umlagert. Bei Commodore hieß es, man könne gegen Ende des Jahres mit der Auslieferung des C 16 und des Plus/4 rechnen.

Die größeren Commodore-Computer waren hauptsächlich mit Steuerungssoftware, Textverarbeitungs- und Anwendungsprogrammen vertreten. Besonders zu erwähnen ist die Textverarbeitung mit vollautomatischer Silbentrennung für die Commodore Serie 600/700 und 8000 von Hard + Soft in Bayreuth.

Auch der neue Commodore PC, der kompatibel zum IBM PC ist, wurde sehr intensiv auf dem Commodore-Stand vorgeführt.

Der eigentliche Favorit aber war — wie immer — der C 64.

Das Hauptangebot bei der Software bestand in Spielen, bei denen es einige Neuvorstellungen gab.



Video- und Akustikdigitalisierer im Einsatz

Ariola führte das Ballerspiel »River Raid« vor. Obwohl Ariolasoft gute neue Programme hat (Seven Cities of Gold, Archon II, Realm of Impossibility), konnten diese leider nicht gezeigt werden. Sie waren wohl aus Versehen von der Hifi-Video-Messe nicht zur Commodore Fachausstellung (CFA) nach Frankfurt, sondern zurück zu Ariola nach München geschickt worden.

Kingsoft stellte zwei neue Spiele vor: »Tom« und »Zaga«, beide durch gute Grafik und günstigen Preis (39 Mark auf Diskette) überzeugend. »Tom« ist ein Spiel, das man zu den intelligenten »Irrgarten-Hüpf-und-Kletterspielen« rechnen kann. »Zaga« ist ein Hubschrauberspiel, das sehr viel Ähnlichkeit zum bekannten »Zaxxon« aufzeigt. Beide Spiele sind voraussichtlich bei Erscheinen dieser Ausgabe schon lieferbar.

Die englische Softwarefirma Anirog brachte gleich drei neue Spiele auf den Markt: »The Soul Gem of Martek«, »P.G. Fuzz« und »The Catacombs«, die ebenfalls durch Leistung und einem Preis von weniger als 40 Mark beeindrucken und ab Oktober über Kingsoft und

Micro-Händler beziehbar sein werden. Bei diesen Spielen zeigte sich schon deutlich der Trend weg von den Schießspielen hin zu den intelligenteren Programmen.

Anspruchslos aber billig waren die Spiele von Bubble Bus: Messepreis 20 Mark das Stück, und das war für die Qualität der Spiele noch zu hoch.

PSS-Software aus England stellte das neue Taktikspiel »Battle for Midway« vor, das



Der »Plus/4« war eine der großen Attraktionen

aus 90% Taktik und 10% Action (Kampf nur bei Konfrontation mit feindlichen Truppen) besteht.

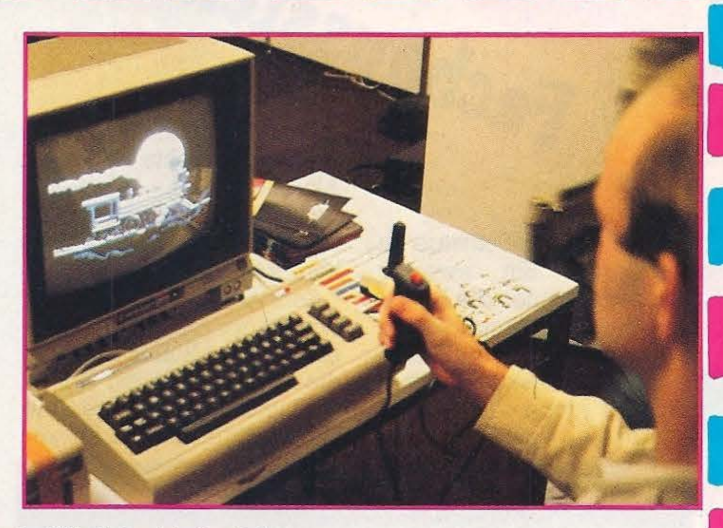
Bei Lucius-Computer-Programme waren zwei Keyboards zu bewundern: Das Colortone-Sensortasten-Keyboard und das Colortone-Professional-Keyboard. Mit dem Sensortasten-Keyboard wird eine ausgezeichnete Musiksoftware geliefert, die wegen Ihrer einfachen Handhabung speziell für Musikanfänger geeignet ist. Zum Professional-Keyboard wird entsprechende Professional-Software geliefert, die von den musikalischen Möglichkeiten her fast mit Musicalc konkurrieren kann. Beide Keyboards sind außerdem noch passend zum Synthesizer-Softwarepaket »Musicalc«, welches ebenfalls von Lucius vertrieben wird.

Das Wersi-Keyboard wurde bereits in der letzen Ausgabe vorgestellt. Wersi setzt mittlerweile den C 64 auch zur Steuerung ihrer großen Heimorgeln ein.

Interface Age zeigte das »Extended Graphik System«, ein gutes Grafik-Utility, über das wir in diesem Heft auch näher berichten.

Handic-Software bot keine neuen Produkte für den C 64, zeigte aber überraschenderweise schon Spiele für den Plus/4: Drei Textadventures, die an die Abenteuerspiele von Scott Adams angelehnt sind, sowie ein einfaches Datenverwaltungsprogramm.

Auch an Hardware wurden einige interessante Neuigkeiten gezeigt, wenn auch die Erwartungen etwas höher ausgefallen waren. Hallberg Elektronik stellte eine 128 KByte-EPROM-Platine für den C 64 vor. Sie ist per Software ein-und ausschaltbar. Damit ist es nun möglich, eine ganze Reihe von verschiedenen Programmen auf EPROM zu brennen und zu nutzen, ohne jedesmal von Diskette zu laden. Sofort beim Einschalten des C 64 stehen alle Programme zur Verfügung. Durch einen POKE-Befehl oder mit den Funktionstasten können Sie das gewünschte Programm aktivieren. Wenn die Karte ausgeschaltet ist, belegt sie weder Speicherplatz noch stört sonst irgendwie. Es können EPROM-Typen von 4 bis 16 KByte eingesteckt werden. Sehr interessant war ein völlig neuartiger Joystick, ebenfalls von Hallberg. Bei diesem Joystick scheint der untere Teil zu fehlen. Er besteht also nur aus dem Griff mit den Feuerknöpfen. Das Prinzip ist einfach und genial zugleich: es wird nach den Gesetzen der Trägheit mit Quecksilber gearbeitet. Zur Steuerung benötigt man nur eine Hand, die andere bleibt frei.

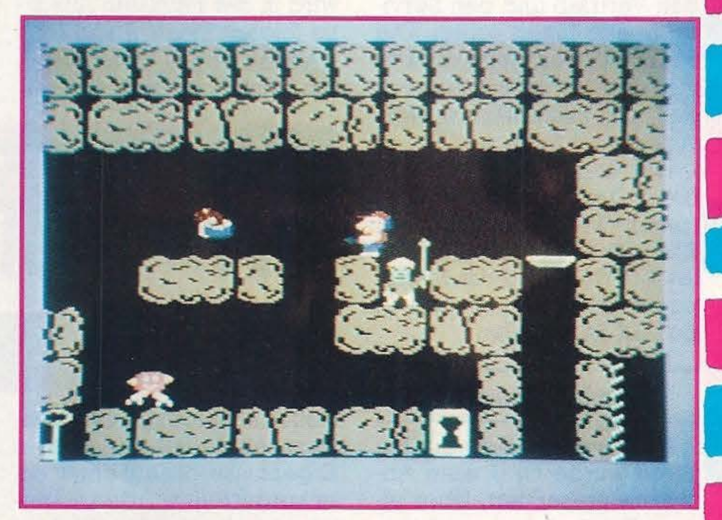


Verblüffend. Joystick ohne Boden

Es wird aber bereits an einer Version gearbeitet, die ähnlich Navigationsinstrumenten in der Schiff- und Luftfahrt mit Kreisel arbeiten, so daß auch Relativ-Bewegungen möglich sind. Das heißt, daß die Bewegung des Cursors (des Sprites, etc.) aufhört, wenn auch der Joystick still steht.

Michael Lamm, ein Hardware-Entwickler, stellte einen neuen IEEE-Bus vor. Durch dessen externes, erweitertes Betriebssystem wird kein Speicherplatz verbraucht. Im Gegensatz zu vielen herkömmlichen IEEE-Bussen sind die RS232-Routinen noch voll erhalten. Die serielle Schnittstelle des C 64 kann ebenfalls benutzt werden. Die Karte wird von Jann-Datentechnik Berlin vertrieben.

Print-Technik und Roland Köhler stellten einen Digitizer vor. Mit Ihnen kann ein von einem Video-Recorder beziehungsweise einer Video-Kamera aufgenommenes Bild in digitale Informationen umgewandelt werden. Diese Bilder lassen sich auf Diskette speichern und von Grafikdruckern ausgeben. Auch Schwingungen, zum Beispiel Töne über ein Mikrofon aufgenommen, können digitalisiert und als grafische Darstellung der Schwingungen verarbeitet werden.

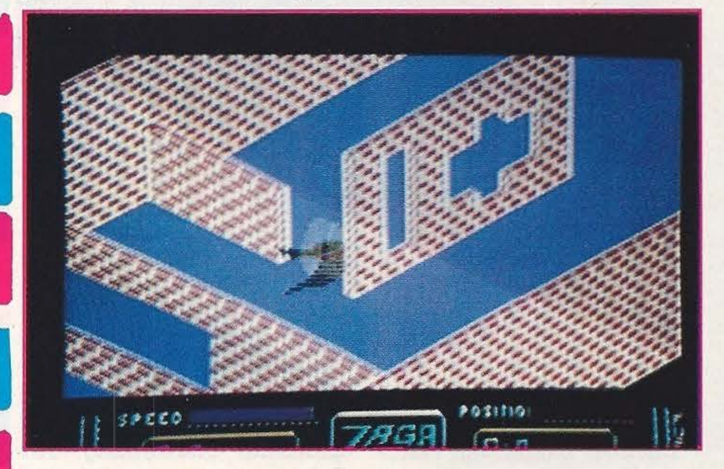


Spielszene aus »Tom«

International Computing aus Holland präsentierte eine wesentlich schnellere VC 1541, bei der alle Datenübertragungen beschleunigt werden. Zu diesem Zweck sind einige Umbauten am C 64 und an der Floppy notwendig. Die Kompatibilität bleibt jedoch voll erhalten. Es wird zusätzlich zu dem seriellen Port eine parallele Datenübertragung installiert. Mit einem Schalter kann umgeschaltet werden. Es ist auch möglich, ein

Commodore Fachausstellung

EPROM mit Btx-Software zu integrieren. Ein weiterer Schalter läßt auch die Benutzung der Diskettenrückseite zu, ohne eine zusätzliche Kerbe in die Diskette zu stanzen. Die junge Firma steht gerade in Verhandlungen



Das Fluggebiet von »Zaga«

mit deutschen Firmen, die den Vertrieb und den Service übernehmen sollen. Die 64'er Redaktion fand die Demonstration so interessant, daß ein ausführlicher Test vorbereitet wird.

Wer seinen SX 64 mit einem Doppellaufwerk ausstatten will, kann dies bei RMC für 2500 Mark machen lassen. Dort wird die Micropower 2000 zerlegt und in den SX 64 eingebaut.

Datenfernübertragung

Am Stand von Software Express wurde DFÜ »zum Anfassen« präsentiert. Über einige Telefonleitungen konnten die Messebesucher Akustikkoppler und Treibersoftware (Teleterm für 198 Mark) ausprobieren. Für diejenigen, die keinen Platz am Computer erreichten, wurde das, was auf einem Monitor zu sehen war, auf eine 2 x 3 Meter große Leuchtdiodenwand übertragen.

Als Renner stellte sich der Akustikkoppler AK 300 (Vertrieb Software Express: Preis 548 Mark) heraus. Er war nach kurzer Zeit vergriffen.

Auch Roboter waren auf diesem Stand zu sehen. Einer von ihnen machte aus einem Akustikkoppler ein »Modem«, in dem er das automatische Abheben des Telefonhörers übernahm.

Sehr interessant war der Workshop vom »Commodore-Guru« Jim Butterfield über die neuen Computer Plus/4, C 16 und C116. Ein Interview mit Jim Butterfield zu diesem und noch anderen Themen wird in der nächsten Ausgabe folgen.

Nach langer Zeit haben die deutschen Softwarehersteller endlich die Lernsoftware entdeckt. Die ersten Ergebnisse, die auf der Messe vorgestellt wurden, konnten uns aber nicht in jedem Fall überzeugen. Man wird noch lernen müssen, daß schnelles Reagieren auf einen neuen Markt, nicht auf Kosten der Qualität gehen darf.

Das Produktspektrum konnte sich allerdings schon sehen lassen.

An die Jüngsten unter den Computer-Benutzern wendet sich der Otto Maier Verlag (Ravensburger) mit Spielen, die das gestalterische Denken entwickeln und fördern sollen. Die Altersgruppen, auf die diese Programme abgestimmt sind, liegen zwischen fünf und zehn Jahren. Die Preise für die neuen Produkte liegen zwischen 88 und 98 Mark.



Titelbild von »Catacombs«

Mathematikprogramme wurden von zwei Anbietern vorgestellt. Der Westermann Verlag beschäftigte sich hauptsächlich mit den Grundrechenarten. Weiterhin wird ein Programm zum Rechtschreibtraining angeboten. Die Produktpalette umfaßt zur Zeit elf Programme, soll aber in Kürze auf 28 Programme erweitert werden. Die Preise hierfür liegen zwischen 39 und 89 Mark. Das Mathematikprogramm ALI von Heureka-Software ist auf die Schüler der Klasse fünf bis hin zur Oberstufe abgestimmt.



▲ Color-Professional-Keyboard

▲ Das Sensortasten-Keyboard von Lucius

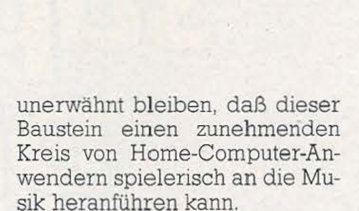
Das größte Angebot ist bei den Vokabellernprogrammen zu finden.

Die Langenscheidt Verlagsgruppe bietet zur Zeit vier Programme für die englische Sprache für je 80 Mark. Die Produktpalette soll in Zukunft erheblich erweitert werden. Also nach dem reinen Übersetzungscomputer von Langenscheidt nun auch Sprachprogramme für Heimcomputer. SM Software hat in seinem Angebot drei Englisch-, zwei Französisch-, einen Spanisch- und einen Italienischkurs. Diese Kurse kosten je 198 Mark. Ein Vokabelprogramm für Latein wird vom Markt & Technik Verlag angeboten.

Einige der Vokabellernprogramme enthalten kleine Spiele. Hat man eine bestimmte Anzahl von Vokabeln gewußt, so darf man als Belohnung einmal spielen.

Ein Lernprogramm ganz anderer Art wurde von Homesoft vorgestellt. Diese Kurse sollen die Computerhändler in die Lage versetzen, ihren Kunden ausreichend Hilfestellung zu geben. Der Einsteigerkurs von 4 x 2 Stunden kostet 98 Mark, der Preis des Basic-Fortgeschrittenenkurses liegt bei 198 Mark. Beide Preise sind inklusive komplettem Unterrichtsmaterials.

(aa/gk/rg/M.Kohlen)



MPS 801: Verständnis für den Handel

Zum Artikel MPS 801 in Ausgabe 8/84

Ich arbeite als Verkaufsberater und Filialleiter in einem Computershop einer Spezialfirma in einem Warenhaus (eingemietet).

Zu dem »...ist bestellt« und »...ist momentan nicht vorrätig« möchte ich Ihnen folgendes mitteilen:
1) Stimmt, kommt vor.
2) Sobald etwas angekündigt ist, wird es bestellt!
3) Die Auslieferung eines Artikels erfolgt meist einige Monate später als die Werbung dafür!
4) Wir können nicht hexen!
5) Die Nachlieferfrist beträgt je nach Hersteller und Artikel ebenfalls bis zu 2 Monaten! (Wenn überhaupt lieferbar!)
6) Suchen Sie den Fehler bezüglich Liefersituation nicht immer im Handel; reklamieren Sie einmal beim Hersteller oder Importeur!
7) Beim momentanen Preissturz kann es sich keine (Klein-) Firma mehr leisten, ein großes Lager zu halten, will sie überleben!
8) Ein wirkliches Fachgeschäft kann bald nicht mehr bestehen, weil die Billigdiscounter die Preise zerstören! (Gutes Personal aus der Computerbranche ist teuer und selten!) Bei uns sagt man »den Bazen und das Weggli kann man nicht haben!« dies heißt: Das Geld und das Brötchen kann man nicht haben!

Übrigens: Software wie div. Spiele können wir wegen Raubkopien meistens nur noch auf Bestellung liefern. Bei uns kauft beinahe niemand mehr Software (Anwenderprogramme und Spiele).

Das Beste: Kommt doch einer und fragt: »Haben Sie auch Spiele?« Ich: »Ja, dieses Regal ist nur für den C 64.« Er: »Nein nein, nicht zum Kaufen, gratis, zum Tauschen!«

Ich hoffe, Sie haben nun auch ein wenig Verständnis für die Seite des Handels gefunden.

Alfred Theiler

Fragen Sie doch!

Selbst bei sorgfältiger Lektüre von Handbüchern und Programmbeschreibungen bleiben beim Anwender immer wieder Fragen offen. Viel mehr Fragen ergeben sich bei Computer-Interessenten, die noch keine festen Kontakte zu Händlern, Herstellern oder Computerclubs haben. Sie können der Redaktion Ihre Fragen schreiben oder Probleme schildern (am einfachsten auf der beigehefteten Karte). Wir veranlassen, daß die Fragen von einem Fachmann beantwortet werden. Allgemein interessierende Fragen und Antworten werden veröffentlicht.

Ziemlich übertrieben!

Im Bericht über die Customer Electronics Show in Chicago in Ausgabe 8/84, Seite 13 steht:

»Diese Musikprogramme sind mittlerweile so ausgereift, daß fast kein Unterschied mehr zu einem digitalen Synthesizer bemerkbar ist.«

Ich halte diesen Absatz für ziemlich übertrieben, ja eigentlich ist dieser sogar falsch! Denn keine noch so ausgefeilte Programmierung macht aus dem digital gesteuerten, aber ansonsten analogen SID des C 64 einen digitalen Sythesizer á la Emulator, Fairlight oder PPG.

Als einziger Vergleichsmaßstab käme eventuell die Bedienungsfreundlichkeit in Frage, mehr aber auch nicht. Auch hier gilt: Hardware kann nicht durch noch so gute Software ersetzt werden.

Abgesehen vom Einsatz als preiswerter Sprachgenerator oder der Eignung zum Heranführen an eine einfache Synthesizer-Programmierung, sollte der SID das bleiben, was er ist, nämlich eine nicht unerhebliche Aufwertung von Computerspielen.

Es ist schade, daß die Begriffe analoge- und digitale Synthesizer in einen Topf geworfen werden, obwohl sich beide Systeme doch ziemlich unterscheiden. Ich möchte kurz folgende Unterscheidungsmöglichkeit aufzeigen:

Bei analogen Synthesizern wird zunächst eine Grundschwingung erzeugt (VCO, DCO). Diese wird über einen steuerbaren Filter (VCF) geleitet und steht hinter einem ebenfalls steuerbaren Verstärker (VCA) zur Verfügung. Anders bei digitalen Synthezisern. Hier wird das Ausgangssignal direkt erzeugt und über D-A-Wandler ausgegeben. So ist es sicherlich zu verstehen, daß Naturklangspeicher-, FM-Synthese- oder rechnende Klang-Parameter-Systeme nicht mit dem Synthesizer-Chip im C 64 zu vergleichen sind. Meiner Ansicht nach wird der SID gelegentlich überbewertet. Es soll allerdings nicht unerwähnt bleiben, daß dieser Baustein einen zunehmenden Kreis von Home-Computer-Anwendern spielerisch an die Musik heranführen kann.

Markus Cohnen
Mitglied im Arbeitskreis Musikelektronik (AME)

VC 20 als 64'er?

Kann man auf dem VC 20 mit 64 KByte Erweiterung auch Programme für den C 64 laufen lassen?
Jan Wilbert

Leider nein, die Hardware beider Computer ist völlig unterschiedlich. Nur Programme ohne PEEK, POKE, SYS und USR laufen einwandfrei.

Modulprogramme laden?

Kann man beim VC 20 Modulprogramme von Diskette laden?
Andreas Hübert

Ja, mit LOAD »NAME«, 8.1. Sie benötigen allerdings RAM im Bereich $A000-$BFFF.

Nochmal MPS 801: Es geht auch einfacher

In bezug auf den Artikel über den MPS 801-Drucker in Ausgabe 8/84 möchte ich Ihnen mitteilen, daß man sich bloß kein komplett neues Farbband kaufen sollte. Den Farbtank kann man auf einfachste Weise mit schwarzer Stempelfarbe nachfüllen! Erst bei starker mechanischer Abnutzung des Farbbandes würde ich ein neues Band empfehlen, da dieses recht teuer ist (25 Mark).

Etwaige Bedenken einer Verstopfung des Druckkopfes sind unbegründet. Ich habe den Tank schon öfter nachgefüllt — ohne Probleme.

Übrigens:
Sperrschrift und Reversschrift sind auch im Direktmodus möglich:

```
OPEN 1,4
PRINT #1, CHR$(14) (bzw. 18 für rvs)
CMD 1
LIST
PRINT #1, CHR$(15) (Normalschrift)
CLOSE 1
```

Hartmut Wenzel

Wollen Sie antworten?

Wir veröffentlichen auf dieser Seite auch Fragen, die sich nicht ohne weiteres anhand eines gutes Archivs oder aufgrund der Sachkunde eines Herstellers beziehungsweise Programmierers beantworten lassen. Das ist vor allem der Fall, wenn es um bestimmte Erfahrungen geht oder um die Suche nach speziellen Programmen beziehungsweise Produkten. Wenn Sie eine Antwort auf eine hier veröffentlichte Frage wissen — oder eine andere, bessere Antwort als die hier gelesene — dann schreiben Sie uns doch. Die Antworten werden wir in einer der nächsten Ausgaben publizieren. Bei Bedarf stellen wir auch den Kontakt zwischen Lesern her.



MSD-Super-Disk-Drive

Bei den Commodore-kompatiblen Einzellaufwerken gibt es inzwischen einige Alternativen. Jetzt gerät allmählich auch der Markt für Doppellaufwerke in Bewegung.

Das »MSD Super Disk Drive«, kurz SD-2, kommt aus den USA und wird durchaus höheren Ansprüchen gerecht.

Das SD-2 kann zwischen dem C-1541-Laufwerk und dem CBM 4040 angesiedelt werden.

Denn die 1541 bietet für eine ganze Menge Anwender zu wenig Leistung, die großen CBM-Floppys der 8000er-Serie sind ganz und gar nicht kompatibel, und das CBM 4040-Doppellaufwerk ist viel zu teuer (neu 2500 bis 3000 Mark).

Komfort-Laufwerk mit guter Optik

Das Gerät überrascht angenehm: Statt einer grauen 1541-Plastik findet man ein sehr solides weißes Metallgehäuse vor, in dem zwei Laufwerke platzsparend verteilt sind. Das MSD-SD-2 nimmt mit beiden Laufwerken nicht sehr viel mehr Platz weg als die C 1541.

Die Disketten werden senkrecht in die Laufwerke eingeführt. Für jedes Laufwerk wird der Betriebszustand einzeln angezeigt. Pro Laufwerk existiert ein Fehlerlämpchen und ein Betriebslämpchen.

Auf der Rückseite des Gerätes befinden sich zwei serielle Anschlüsse und ein paralleler IEEE-488-Bus, der Netzanschluß, ein Ein/Aus-Schalter sowie eine gut zugängliche Sicherung.

Nicht schnell wie der Wind, aber flink wie ein Wiesel

Ein gutes Argument für den Kauf des MSD-Doppellaufwerks ist die Geschwindigkeit, die sich je nach Schnittstelle richtet.

Wählt man die serielle Schnittstelle, so ist die Geschwindigkeit annähernd die gleiche wie bei der 1541 (sofern man hier von Geschwindigkeit reden kann), sogar minimal langsamer.

Lohnender ist allerdings der Anschluß über den IEC-Bus, zu dem man allerdings ein spezielles Interface kaufen muß. Das Laufwerk wurde mit den IEEE-488-Interfaces von Telesys, Dynamics und RMC-Systems getestet.

Die Geschwindigkeitsvergleichstests wurden anhand eines 153 Block langen Files durchgeführt (siehe Tabelle).

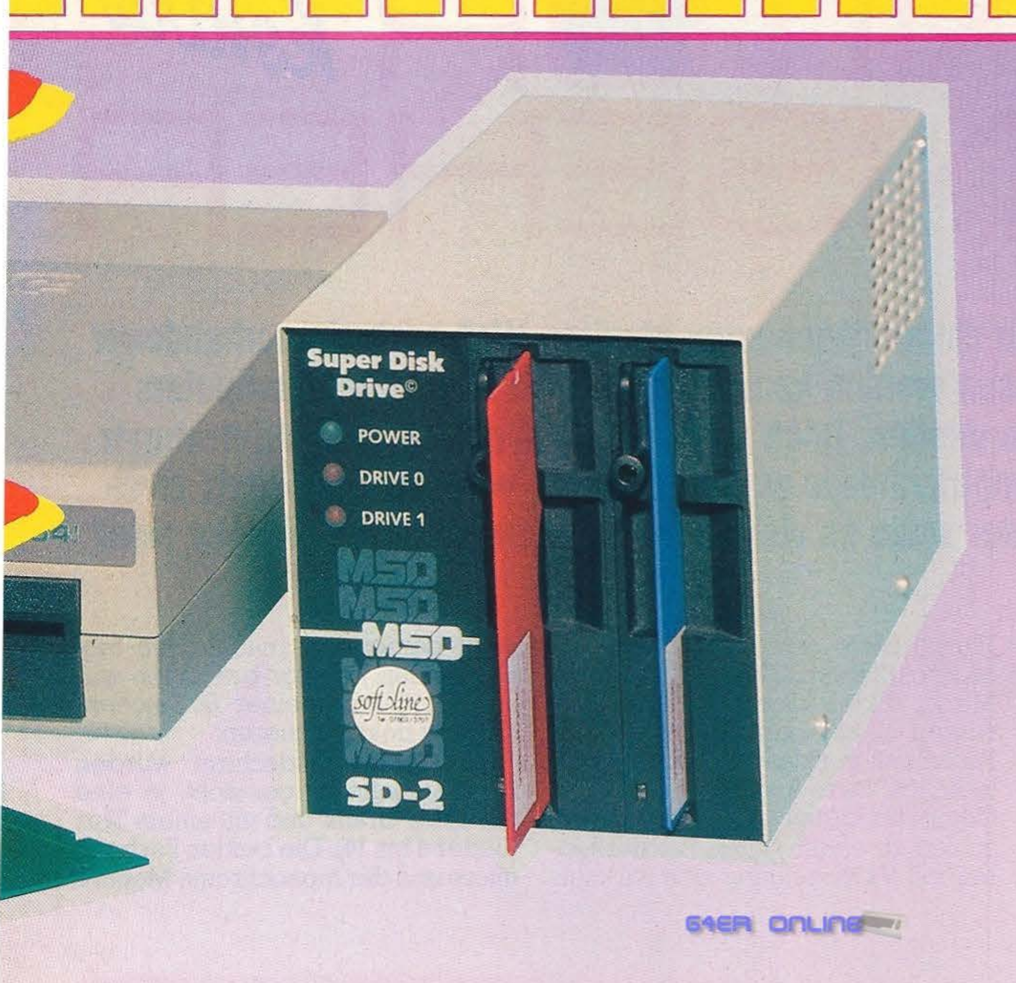
Das Kopieren einer ganzen Disk benötigt auf dem MSD-Floppy nur etwa knapp 2 Minuten: Einfach Original- und Zieldiskette in die Drives und Backup-Befehl ans Floppy senden. Schade, daß man mit einem Einzellaufwerk wie der 1541 nicht so einfache Befehle wie Backup ("D1=0") oder Copy ("C0=1") benutzen kann.

Kompatibel muß es sein

Obwohl die MSD-Floppy also nicht die gleiche Geschwindigkeit herzaubert wie das CBM-4040-Doppellaufwerk (über IEC-Bus zirka fünfmal schneller), ist das SD-2 mit dem Geschwindigkeitsfaktor 2,3 mal schneller und im Hinblick auf die Kompatibilität der Software mehr zu empfehlen.

Diese Unterschiede lassen sich mit den Eigenarten der Floppy-Betriebssysteme erklären, die mit der 1541 natürlich nicht identisch sind und aus Copyright-Gründen auch gar nicht sein dürfen.

Das MSD-Laufwerk ist mit seriellem Kabel zu zirka 90% der Software kompatibel, sogar der doch etwas kompliziertere Disketten-Kopierschutz von Synapse-Software lief darauf. Bei der Wahl der parallelen



Schnittstelle sieht die Sache leider etwas düsterer aus:

Nur etwa 75% der kommerziellen, kopiergeschützten Software lief darauf.

Hätte MSD nicht aus den Fehlern der alten CBM-4040-Floppy gelernt, wäre die Kompatibilität zum C-1541-Einzellaufwerk gänzlich in den Wind zu schreiben gewesen. Bei vielen Software-Produkten wird keine Laufwerksnummer 0 oder 1 angegeben. Das 4040-Floppy hat die Angewohnheit, dabei immer auf beide Laufwerke zuzugreifen und Programme wie zum Beispiel die Adventurespiele von Infocom nur zu starten, wenn in beiden Laufwerken die gleiche Disk vorhanden ist. Das MSD-Laufwerk greift automatisch auf Laufwerk 0 zu, um die Kompatibilität zum Einzellaufwerk stärker zu gewährleisten.

Solide Hardware bis ins letzte Detail

Bei der Verarbeitung zeigt das MSD Super-Disk-Drive große Vorteile gegenüber dem C-1541-Laufwerk. Zum einen ist es bei weitem nicht so überhitzungsgefährdet wie das Commodore-1541, zum anderen ist es sehr stabil gebaut. Nach einem Härtetest (25 Disketten hintereinander formatieren) hatte sich der Schreib/Lesekopf nicht einmal um Bruchteile von Millimetern verschoben — der Schreib/Lesekopf des 1541 wäre längst dejustiert gewesen. Bemerkenswert ist auch, daß bei auftretenden Lesefehlern kein lautes Rattern wie am 1541 oder 4040 zu hören ist. Das SD-2 schont sich also praktisch selbst.

Die besondere Fähigkeit

Laut englischer Anleitung weist das MSD-Super-Disk-Drive einen besonderen Gag auf: Nachdem die Datendiskette im einen Laufwerk voll ist, wird einfach auf eine Diskette im 2. Laufwerk umgeschaltet. Dies würde natürlich für den kommerziellen Anwender bei vollen 340 KByte Speicherkapazität sehr von Nutzen sein. Leider lag uns zum Test noch keine Version des MSD-Doppel-Disk-Drives vor, das diese Umschaltung geboten hätte.

Alles in allem gesehen, kann man das SD-2 als hervorragendes Gerät bezeichnen, das sich besonders durch seine stabile Verarbeitung auszeichnet. 1998 Mark für dieses Diskettenlaufwerk (ohne IEC-Bus) sind kein besonders billiges Vergnügen. Es lohnt sich aber trotzdem für die kommerziellen Anwender und Viel-Kopierer.

Für Profis geeignet

Insbesondere die üblichen Reparaturkosten, die man sich mit dem Kauf des 1541 früher oder später aufhalst, sprechen für den Kauf des MSD-Laufwerks.

Dem einfachen Hobby-Benutzer, der seinen C 64 (plus Floppy) nicht länger als drei Stunden pro Tag angeschaltet hat (echte Dauerbenutzer haben mindestens ein Pensum von fünf Stunden täglich), ist jedoch weiterhin der Kauf der sehr viel billigeren C 1541 anzuraten, besonders im Hinblick auf die Kompatibilität der Software. (M. Kohlen)

Info: Softline

	Laden	Copy-Befehl	Back-up-Befehl
1541	97 s	nicht vorhanden	nicht vorhanden
serielles MSD	108 s	111 s	115 s
paralleles MSD	46 s	111 s	115 s
CBM 4040	23 s	105 s	116 s

Tabelle. Geschwindigkeitsvergleich anhand eines 153 Block langen Files

Höhe 15,7 cm
Breite 15 cm
Tiefe 33,8 cm
Anschlüsse: wahlweise IEEE-Bus parallel oder seriell.
Diskformat: 5,25-Zoll-Disketten Single Density
Speicherkapazität:
pro Laufwerk wie 1541:
sequentiell 168 656 Byte
relativ 176 132 Byte

Einige Daten zum MSD-SD-2

MONITOR *kontra* FERNSEHER

Nur wenige Computer haben einen eingebauten Monitor. Weil auch ein vorhandener Fernsehapparat angeschlossen werden kann, können die Preise eines Computers auch relativ niedrig gehalten werden. Doch wenn man es leid ist, das Wohnzimmer mit einer undekorativen Computeranlage zu verunstalten, wird der Kauf eines zweiten Fernsehers erwogen. Oder sollte es vielleicht besser ein spezieller Monitor sein?

Bildschirmgeräte stehen heute in fast allen Haushalten — in Form eines Fernsehgerätes. Kein Wunder also, wenn viele Hersteller von Heimcomputern auf das vorhandene Reservoir zurückgreifen und für ihre Geräte ein solches Fernsehgerät als Ausgabeeinheit vorsehen. In manchem Heimcomputer-Besitzer mag aber — nach langen Abenden vor dem zweckentfremdeten Fernseher — der Wunsch entstanden sein, auch einmal vor einem scharfen Monitorbildschirm sitzen zu dürfen.

Wir haben uns fünf Geräte ausgesucht und von jedem Bildschirm einige Aufnahmen gemacht. Wir wählten zwei Farbmonitore aus (das Modell 1701 von Commodore (zirka 900 Mark) und den Taxan Vision Ex (zirka 1000 Mark)), zwei Farbfernseher mit Video-Eingang (Panasonic TC 1637 DR (zirka 1000 bis 1100 Mark) und Sharp DV-1400 G,S,N (zirka 900 Mark)) sowie einen monochromen Monitor (Zenith ZVM-123-E (zirka 350 Mark), Bilder 1 bis 3). Wir wollten wissen, ob es lohnend ist, sich einen Farbmonitor anzuschaffen, der sich lediglich zum Anschluß an einen Computer eignet und ob der Unterschied zu einem guten Farbfernseher mit Videoeingang sich wirklich stark bemerkbar macht. Den monochromen Monitor brauchten wir, um die Qualität einer 80-Zeichen-Karte zu dokumentieren.

Von jedem Bildschirm wurden zwei Aufnahmen gemacht, je eine mit einer Grafik und mit einem Text (Bilder 4 bis 14). Die beiden Farbmonitore und der monochrome Monitor mußten sich zusätzlich noch mit der 80-Zeichen-Karte quälen lassen (Bilder 15 bis 17). Sie stammt von der Firma Decam electronic aus Ettlingen. Bei den Aufnahmen muß man berücksichtigen, daß die Einstellungen von Farbe, Helligkeit und Kontrast sich verändern lassen, daß auch foto- und drucktechnische Gegebenheiten das Ergebnis etwas verfälschen können. Doch lassen sich einige grundsätzliche Unterschiede schon feststellen.

Um die Unterschiede noch etwas klarer herauszustellen, haben wir zu jedem Bildschirm eine Ausschnitts-



Bild 1. Farbmonitore Taxan Vison EX und Commodore Modell 1701

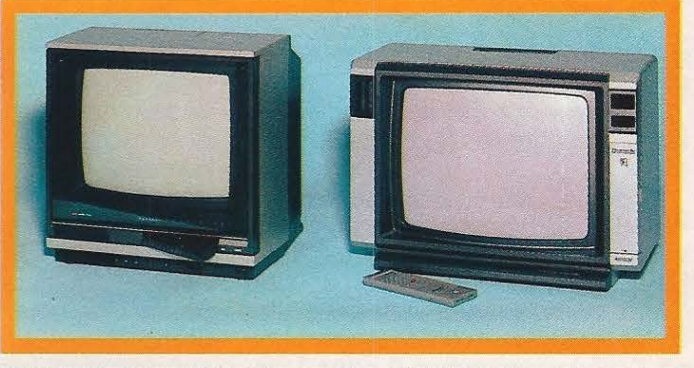


Bild 2. Farbfernseher mit Video-Eingang: Sharp DV-1400 G,S,N und Panasonic TC 1637 DR

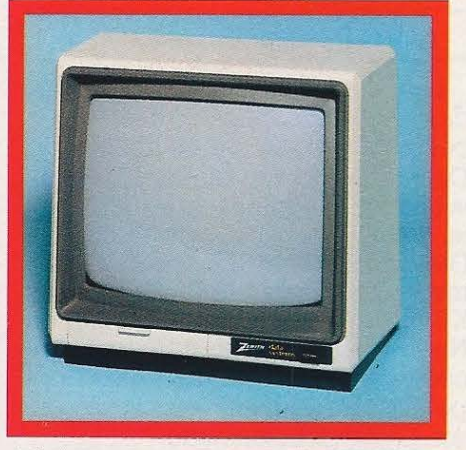


Bild 3. Monochromer Monitor Zenith ZVM-123-E

vergrößerung der Grafik (Bild 4) und des Textes gewählt. Hier wird auch die Struktur des Bildschirmes sichtbar.

Da die Sehgewohnheiten sehr unterschiedlich sind, haben wir auf eine Bewertung verzichtet. Dieser Artikel ist auch kein Vergleichstest, sondern soll Ihnen die Unterschiede zwischen Fernseher und Monitoren demonstrieren.

»Zeichen pro Zeile« ist Unsinn

Viele Anwender interessieren sich im Zusammenhang mit Textverarbeitung mit den Problemen bei der Darstellung von einer bestimmten Anzahl von Zeichen pro Zeile.



Bild 4. Diese Grafik wurde als Vergleich herangezogen. Die folgenen Bilder zeigen einen Ausschnitt



Bild 5. Text auf dem Panasonic



Bild 6. Ausschnittvergrößerung von Bild 4 am Farbfernseher Panasonic



Bild 7. Text auf dem Sharp



Bild 8. Bildschirmausschnitt des Sharp Farbfernsehers

Die Angabe »Zeichen pro Zeile« hat sich eigentlich erst verbreitet, als sich zunehmend technisch Unbedarfte im Computerbereich zu interessieren begannen. Sie scheint so schön anschaulich und unkompliziert zu sein. Dabei ist sie Ursache einer großen Verwirrung in diesem Bereich. Sie suggeriert nämlich, daß es im Monitor irgendeine Stelle gäbe, die entweder 40 oder 80 Zeichen akzeptieren würde, aber nicht beides und nichts zwischendrin. In einigen Anzeigen wird dieses Mißverständnis noch gefördert, indem sie von »umschaltbar auf 40 oder 80 Zeichen« sprechen!

Der Monitor erkennt Zeichen schon aus technischen Gründen überhaupt nicht als solche, da er ja immer nur die auf einer Bildzeile nebeneinanderliegenden Matrixzeilen einer ganzen Textzeile schreibt. Jede Punktmenge, die kleiner ist als die maximal mögliche, wird selbstverständlich auch abgebildet. Ein Monitor, der 400 Punkte pro Zeile abbilden kann, kann auch 200 abbilden, oder 360 oder irgendeine andere Anzahl, ohne Umschaltung. Ob diese Punkte zu 22, 40 oder 80 Zeichen gehören, ist ihm, schlicht gesagt, egal.

Die Umschaltmöglichkeit, die manche Hersteller anbieten, hat einen anderen Grund: Wenn nämlich ein Monitor, der 800 Punkte pro Bildzeile noch scharf abbildet, aber von einem Computer nur 400 angeboten bekommt, kann die Auflösung zu gut sein. Dann sind die Punkte durch deutliche Zwischenräume getrennt. Das ist unter Umständen für den Betrachter unangenehm. Mit dem Umschalter wird die Bandbreite deshalb absichtlich soweit eingeschränkt, daß die Punkte etwas verschliffen werden. Das Auge nimmt sie breiter gezeichnet wahr, die Zeichen erscheinen im Bild geschlossener und die Arbeit mit ihnen ermüdet weniger. Alle diese Ausführungen treffen in vollem Maße auch auf Farbmonitore zu.

Welches Bildschirmgerät ist aber am geeignetsten? Nun, das kommt ganz auf den Verwendungszweck

an. Für Textverarbeitung und alle Anwendungen, die auf die Darstellung großer Datenmengen pro Bildschirmzeile beruhen, erst recht bei 80-Zeichen-Karten, ist ein monochromer Bildschirm am besten geeignet, es sei denn, daß die Farbe ein wesentlicher Bestandteil der Bildinformation ist. Selbst gute Farbmonitore haben aber nur eine Auflösung, die derjenigen billiger monochromer Monitore entspricht. Absolut ungeeignet ist ein Farb- oder Schwarzweiß-Fernseher für diesen Zweck. Für grafische Anwendungen bietet sich ein Farbmonitor ab mittlerer Preislage an. Eventuell geht auch ein Farbfernseher mit Video-Eingang; die Farben sind zwar dann brillant, an der Auflösung mangelt es jedoch. Die Domäne normaler Farbfernsehgeräte sind eigentlich nur Anwendungen im Bereich der Videospiele. Hierbei ist zum einen der Tonteil von Bedeutung und andererseits keine übermäßige Schärfe nötig.

Insgesamt betrachtet, erscheint ein Farbfernseher mit Videoeingang ein guter Kompromiß zu sein. Man sollte dann jedoch darauf achten, daß die Bildschirmgröße bei 20 bis 35 cm Schirmdiagonale liegt. Dann kann man in einem optimalen Abstand zwischen 0.5 und 1.5 Metern vom Bildschirm sitzen (schon deshalb ist ein großer Farbfernseher ungeeignet). Verbindliche Werte gibt es jedoch nicht. (gk)

Bezugsquellen: siehe Marktübersicht auf Seite 22. Die beiden Farbfernseher sind im Fachhandel erhältlich.

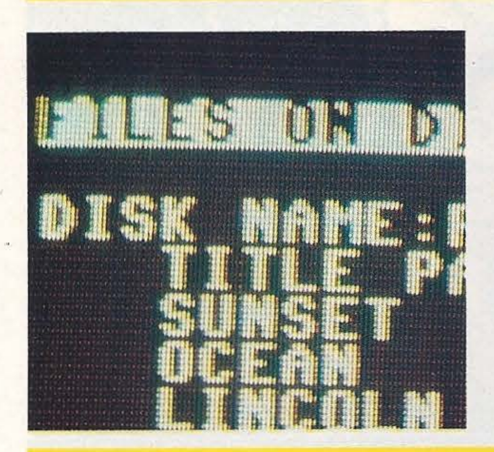


Bild 9. Text auf dem Farbmonitor Taxan Vision Ex


Bild 10. Bildschirmausschnitt vom Taxan Vision Ex

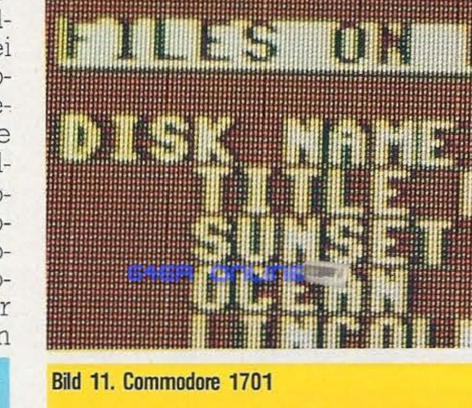


Bild 11. Commodore 1701

Bild 12. Commodore 1701.



Bild 13. Monochromer Monitor Zenith ZVM-123-E


Bild 14. Monochromer Monitor Zenith ZVM-123-E

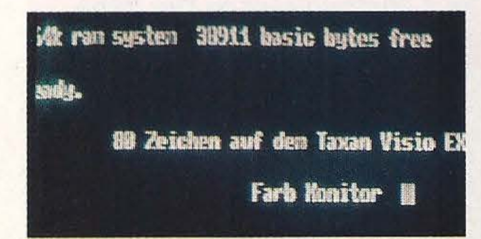


Bild 15. Bildschirmausschnitt bei einer 80-Zeichen-Karte bei dem Farbmonitor Taxan



Bild 16. Text mit einer 80-Zeichen-Karte beim Commodore 1701

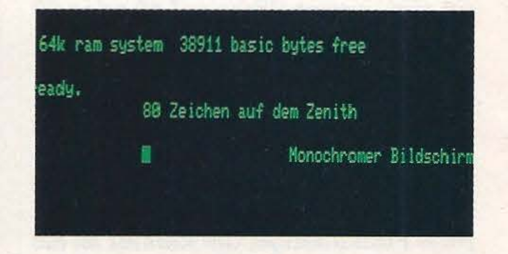


Bild 17. Sehr klares Bild bei Verwendung einer 80 Zeichen-Karte bei dem monochromen Monitor Zenith ZVM-123-E

Marktübersicht Schwarzweiß- und Farbmonitore

Ob wegen Textverarbeitung oder Familienprotest — immer mehr Computerbesitzer ersetzen den Fernseher durch einen Monitor. Diese Marktübersicht soll eine kleine Hilfe im Vorfeld der Kaufentscheidung sein.

Wir haben uns dabei auf die Monitore beschränkt, die vom Preis her in ein C 64-System passen. Das bedeutet im Klartext, daß wir monochrome Monitore oberhalb von etwa 600 Mark ebensowenig berücksichtigt haben wie Farbmonitore, die wesentlich mehr als etwa 1600 Mark kosten.

Die angegebenen Preise verstehen sich als ungefähre Richtwerte; Preisvergleiche bei verschiedenen Händlern lohnen sich in der Regel.

Die meisten Monitore werden über den Fachhandel oder über Kaufhäuser vertrieben. Die folgende Anbieterliste enthält für den Interessierten die Adressen der Vertriebsfirmen, bei denen Datenblätter und nähere Informationen zu bekommen sind. (ev)

Anbieter:

Apple Computer, Freischützstr. 92, 8000 München 81
Commodore Deutschland, Lyoner Str. 38, 6000 Frankfurt 71
Feitron Elektronik, Auf dem Schnellerod 22, 5210 Troisdorf
Gerb Elektronik, Roedernallee 174-176, 1000 Berlin 51
Hantarex Deutschland, Siegener Str. 23, 5230 Altenkirchen
ITT/SEL, Frankstr. 60, 7530 Pforzheim
Loewe Opta, Industriestr. 11, 8640 Kronach
C. Melchers & Co., Schlachte 39/40, 2800 Bremen 1
Microscan, Postfach 601705, 2000 Hamburg 60
Mirwald Electronic, Fasanenstr. 8b, 8025 Unterhaching
National Panasonic, Winsbergring 15, 2000 Hamburg 54
Philips Data Systems, Weidenauer Str. 211-213, 59 Siegen 21
Sanyo Deutschland, Widenmeyerstr. 25, 8000 München 22
Unitronic, Münsterstr. 338, 4000 Düsseldorf 30
Zenith Data Systems, Robert-Bosch-Str. 32-38, 6072 Dreieich-Sprendlingen

Marktübersicht Drucker



Hier ist der zweite Teil unserer aktuellen Marktübersicht über das Angebot an Druckern und Plottern für den Commodore 64 und den VC 20.

Wie bereits im ersten Teil, so fehlen auch diesmal zwei Kategorien von Druckern und Plottern. Zum einen nämlich Geräte aus Preisklassen, die für den C 64/ VC 20-Anwender nicht mehr interessant sind, also beispielsweise Superdrucker für 5000 Mark oder DIN A3-Plotter im fünfstelligen Preisbereich. Zum anderen wurden auch diesmal Drucker nicht berücksichtigt, die nur mit unvertretbar großem Aufwand an den C 64 / VC 20 angeschlossen werden können.

Sie können also davon ausgehen, daß alle in dieser Marktübersicht vertretenen Drucker und Plotter ohne große Probleme mit dem C 64/ VC 20 arbeiten. Bei Geräten, die direkt über den seriellen Bus anschließbar sind, ist in der Rubrik »Interface« »C 64/VC 20« vermerkt.

Die Liste der an den C 64/VC 20 anschließbaren Drucker soll auch zukünftig weiter aktualisiert werden. An dieser Stelle sind daher auch die Anbieter entsprechender Drucker oder Plotter aufgefordert, uns entsprechende Informationen zukommen zu lassen.

Alle Preisangaben sind nur ungefähre Werte. Wer sich bei verschiedenen Anbietern informiert, kann unter Umständen günstiger einkaufen.

Anbieter-Liste Drucker & Plotter (Teil 2)

Die hier aufgeführten Adressen sind in der Regel keine direkten Bezugsquellen. Sie erhalten hier jedoch Datenblatt und Händlernachweis für den von Ihnen ins Auge gefaßten Drucker oder Plotter. (ev)

Ascalon-Vertrieb Mayon Elektronik GmbH, Postfach 1925, 8034 Germering
Brother International GmbH, Im Rosengarten 14, 6368 Bad Vilbel
Centronics System Elektronik Vertriebs GmbH, Heesfeld 4, 3300 Braunschweig
DEC Transtec Computervertriebsges. mbH, Schaffhausenstr. 113, 7400 Tübingen
Dyneer Vertrieb Technitron GmbH, Charles-de-Gaulle-Str. 4, 8000 München 83
Facit-Ericson, Niederheider Str. 3, 4000 Düsseldorf 13
Honeywell-Fachhandel Lackner GmbH, Willstr. 2, 8500 Nürnberg
C. Itoh Deutschland GmbH, Königsallee 21, 4000 Düsseldorf 1
Macrotron, Stahlgruberring 28, 8000 München 82
Mannesmann-Tally GmbH, Postfach 2969, 7900 Ulm
G. Melchers & Co., Schlachte 39/40, 2800 Bremen 1
Mitsui-Vertrieb Technitron GmbH, Charles-de-Gaulle-Str. 4, 8000 München 83
NEC Europa, Wiesenstr. 148, 4040 Neuss 1
Olivetti-Fachhandel, Ingenieurbüro Jörg Michael, St. Katharinen-Weg 6, 7750 Konstanz 16
Roland GmbH, Postfach 1905, 2000 Norderstedt
Siemens Vertrieb PAV, Gerzener Str. 5, 8311 Dietelskirchen
Star Europe GmbH, Frankfurter Allee 1-3, 6236 Eschborn

Marktübersicht Schwarzweiß- und Farbmonitore

Hersteller	Modell	SW/ Farbe	Diagonale (cm)	Schirmfarbe	Auflösung	Entspiegelung	Bandbreite	Signal	Anschlußbuchse	Ton	Zubehör	Preis	Anbieter
Apple	Apple Monitor	SW	31	grün	k.A.	ja	18 MHz	BAS	Cinch	k.A.	k.A.	560,—	Fachhandel
BMC-Mirwald	BM 12 A	SW	31	grün	480 x 240	nein	15 MHz	BAS	Cinch	nein	k.A.	298,—	Mirwald Electronic
BMC-Mirwald	BM 12 EN	SW	31	grün	480 x 240	geätzt	20 MHz	BAS/ FBAS	Cinch	nein	Ergotilt-Sockel	498,—	Mirwald Electronic
BMC-Mirwald	BM 12 EY	SW	31	bernstein	640 Zeilen	Filter	18 MHz	BAS	Cinch	nein	k.A.	440,—	Mirwald Electronic
BMC-Mirwald	BM 8181	Farbe	31	8	640 x 240	geätzt	k.A.	RGB	DIN	k.A.	Schaltplan, Kabel	1600,—	Mirwald Electronic
Commodore	1701	Farbe	33	beliebig	440 x 330	nein	k.A.	BAS/ FBAS	Cinch	ja	Videokabel, Handbuch	900,—	Commodore
Commodore	1702	Farbe	33	beliebig	440 x 330	nein	k.A.	BAS/ FBAS	Cinch	ja	Videokabel, Handbuch	920,—	Commodore
Hantarex	CMT 2000 12"	SW	31	grün	k.A.	Aufpreis	18 MHz	BAS/TTL	BNC	nein	k.A.	435,—	Hantarex Deutschl.
Hantarex	CMT 2000 9"	SW	23	grün	1000 x 625	Aufpreis	18 MHz	BAS/TTL	BNC	nein	k.A.	420,—	Hantarex Deutschl.
Hantarex	CT 2000 15"	SW	38	grün	k.A.	Aufpreis	18 MHz	BAS/TTL	BNC	nein	k.A.	500,—	Hantarex Deutschl.
Loewe Opta	DM 114	Farbe	35	4086 (Btx)	445 x 625	nein	10 MHz	RGB/BAS	Scart	ja	Fuß, Scart-Kabel	1200,—	Fachhandel
Matsushita	JB-3062	SW	31	grün	640 Zeilen	geätzt	18 MHz	BAS	RCA	nein	k.A.	550,—	National Panasonic
Novex	1414-CL	Farbe	36	beliebig	413 x 560	nein	k.A.	RGB/BAS	2 x Cinch	ja	k.A.	1120,—	Gerb Elektronik
Panasonic	BM 3202 G/A	SW	27	grün	k.A.	ja	15 MHz	Video	Cinch	nein	Kabel, Sichtscheibe	900,—	ITT/SEL
Philips	PCT 1201	SW	31	bernstein	k.A.	ja	22 MHz	BAS	Cinch	nein	k.A.	480,—	Fachhandel
Prince	12.10.02	SW	31	grün	k.A.	k.A.	24 MHz	BAS	2 x Cinch	nein	Anschlußkabel	450,—	Unitronic GmbH
Sanyo	CDM 14 RX	Farbe	36	8	k.A.	k.A.	7 MHz	RGB	8-Pin	k.A.	k.A.	1600,—	Sanyo Video Vertrieb
Sanyo	DM 2112	SW	31	grün	k.A.	geätzt	15 MHz	BAS	Cinch	nein	k.A.	320,—	Sanyo Video Vertrieb
Sanyo	DM 2212	SW	31	bernstein	k.A.	geätzt	15 MHz	BAS	Cinch	nein	k.A.	349,—	Sanyo Video Vertrieb
Sanyo	DM 8112	SW	31	grün	k.A.	geätzt	18 MHz	BAS	Cinch	nein	Anschlußkabel	630,—	Feltron
Sanyo	DM 8112 CX	SW	31	grün	k.A.	geätzt	20 MHz	BAS	Cinch	nein	k.A.	520,—	Sanyo Video Vertrieb
Sanyo	DM 8212 CX	SW	31	bernstein	k.A.	geätzt	20 MHz	BAS	Cinch	nein	k.A.	550,—	Sanyo Video Vertrieb
Toei	KH 12 A	SW	31	bernstein	k.A.	nein	20 MHz	BAS	2 x Cinch	nein	Filterscheibe	500,—	Sanyo Video Vertrieb
Toei	KH 12 G	SW	31	grün	k.A.	geätzt	20 MHz	BAS	2 x Cinch	nein	Filterscheibe	500,—	Sanyo Video Vertrieb
Taxan	RGB Vision-Ex	Farbe	31	beliebig	380 x 262	nein	18 MHz	RGB/BAS	k.A.	k.A.	Kontrastscheibe	998,—	C. Melchers & Co.
Taxan	RGB Vision-I	Farbe	31	beliebig	380 x 262	nein	18 MHz	RGB	k.A.	k.A.	Kontrastscheibe	950,—	C. Melchers & Co.
Taxan	RGB Vision-II	Farbe	31	beliebig	510 x 262	nein	18 MHz	RGB	k.A.	k.A.	Kontrastscheibe	1385,—	C. Melchers & Co.
Teco	TM-1265	SW	31	grün	800 Zeilen	geätzt	20 MHz	BAS	Cinch	nein	Monitorkabel	398,—	Microscan
YJE	CM-1411	Farbe	36	beliebig	380 x 300	ja	k.A.	BAS/ FBAS	RCA	k.A.	Monitorkabel	950,—	Microscan
YJE	KD-1410	Farbe	33	16	380 x 240	ja	18 MHz	RGB	Submin. D	k.A.	Videokabel	1500,—	Microscan
Zenith	ZVM 122 EA	SW	31	bernstein	640 x 450	geätzt	18 MHz	BAS	Cinch	nein	k.A.	353,—	Zenith Data Systems
Zenith	ZVM 123 EA	SW	31	grün	640 x 450	geätzt	18 MHz	BAS	Cinch	nein	k.A.	353,—	Zenith Data Systems
Zenith	ZVM 124 E	SW	31	bernstein	720 x 350	ja	22 MHz	TTL	Submin.D	nein	k.A.	855,—	Zenith Data Systems
Zenith	ZVM 133 E	Farbe	33	16	640 x 240	nein	20 MHz	RGB	DIN	ja	Monitorkabel	1700,—	Zenith Data Systems

Marktübersicht Drucker für C 64/VC 20 (Teil 2)

Modell	Typ	Zeichen pro Sekunde	Zeilenlänge	Druck-Matrix	Farben	Papierart	Zeichensatz	Puffer	Grafik	Interface	Bemerkung	Preis	Bezugsquelle
Ascalon SC-1000	Matrix	100	40-132	9 x 11	—	Endlos, Einzelblatt	8 Zeichensätze	k. A.	k. A.	Centronics, RS232C		990,—	Mayon Elektronik
Brother HR-5C	Thermomatrix	30	40/80	9 x 9	—	Einzelblatt, Rolle	Commodore	1 Zeile	480 Punkte pro Zeile	C 64 / VC 20		499,—	Fachhandel, Kaufhäuser
Centronics 154-2	Matrix	120	80	8 x 11	—	Endlos, Einzelblatt	ASCII	2 KByte	k. A.	Centronics		2000,—	Fachhandel
DEC LA-50	Matrix	100	80	8 x 9	—	Endlos	ASCII	k. A.	k. A.	RS232C		2100,—	Fachhandel
Dyneer DW 16	Typenrad	16	80	—	—	Endlos, Einzelblatt	je nach Typenrad	k. A.	—	Centronics, RS232C		1870,—	Fachhandel
Facit 4510	Matrix	120	80	9 x 9	—	Endlos, Einzelblatt	8 Zeichensätze	2 KByte	k. A.	Centronics, RS232C		1750,—	Fachhandel
Honeywell L31CQ	Matrix	100	bis 220	9 x 7	—	Endlos	ASCII	2 KByte	k. A.	Centronics		2200,—	Fachhandel
Itoh CX-4800	Trommelplotter	—	—	—	4	Rolle	ASCII	k. A.	über Plot-Kommandos steuerbar	Centronics	Generatoren für Linien, Kreise, Achsenkreuze	2300,—	Fachhandel
Macrotron Juki 6100	Typenrad	22	k. A.	—	—	Endlos, Einzelblatt	je nach Typenrad	k. A.	—	Centronics, RS232C		2000,—	Fachhandel
Macrotron Speedy-80	Matrix	80	40-142	7 x 8	—	Endlos, Einzelblatt	ASCII	k. A.	bis 1280 Punkte pro Zeile	Centronics, RS232C	VC-Interface gegen Aufpreis	875,—	Fachhandel
Mannesmann MT-160	Matrix	132	80/132	9 x 9	—	Endlos, Einzelblatt	8 Zeichensätze	2 KByte	k. A.	Centronics, RS232C		2550,—	Fachhandel
Melchers CP-80	Matrix	80	80/142	8 x 9	—	Endlos, Einzelblatt, Rolle	8 Zeichensätze	2 KByte	640 Punkte pro Zeile	Centronics, optional RS232C		875,—	Fachhandel
Mitsui MC-2100	Matrix	120	80	9 x 7	—	Endlos, Einzelblatt	4 Zeichensätze	80 Byte	k. A.	Centronics, RS232C		1495,—	Fachhandel
Mitsui MC-2200	Matrix	180	80-136	9 x 9	—	Endlos, Einzelblatt	11 Zeichensätze	2 KByte	k. A.	Centronics, RS232C		1650,—	Fachhandel
NEC Pinwriter P2	Matrix	117	80-137	9 x 9	—	Endlos, Einzelblatt	8 Zeichensätze	k. A.	Bit-Image Grafik	Centronics	freiprogrammierbarer Zeichensatz	2050,—	Fachhandel
Olivetti Praxis 40	Typenrad	k. A.	k. A.	—	—	Einzelblatt	ASCII	k. A.	nein	C 64	Spezialausführung einer elektr. Schreibmaschine	1398,—	Fachhandel
Roland DXY 101	Flachbettplotter	—	—	—	1	Einzelblatt (A4), Folie	ASCII	k. A.	Zeichenfläche 370 x 260 mm	Centronics, RS232C	Generatoren für Kreise, Linien, Achsenkreuze	2000,—	Fachhandel
Siemens PT 89	Matrix	150	bis 267	9 x 9	—	Einzelblatt	8 Zeichensätze	k. A.	k. A.	Centronics		2400,—	Fachhandel
Star Delta 10	Matrix	160	80-136	9 x 9	—	Endlos, Einzelblatt	8 Zeichensätze	k. A.	Bit-Image Modus bis 1920 Punkte pro Zeile	Centronics, RS232C	96 freiprogrammierbare Zeichen	1195,—	Fachhandel

Ein starkes Stück

Die beiden Matrixdrucker Itoh 8510 und 1550B unterscheiden sich hauptsächlich durch ihre Druckbreite. Enorm leistungsfähig sind sie beide.

Der optische Unterschied ist auffällig, denn der Itoh 8510 (Bild 1) ist im Gegensatz zum 1550B ein fast zierlicher Drucker. Dabei haben beide Drucker viele Gemeinsamkeiten. Sie arbeiten nach dem Punktmatrixprinzip mit einer Zeichenmatrix von maximal 8x8 Zeichen. Auch ist ihnen die Fähigkeit sowohl Einzelblatt als auch Endlospapier zu verarbeiten gemeinsam. Der Unterschied liegt in der Papierbreite. Der Itoh 1550B druckt mit einer Breite bis zu 15 Inch, der Itoh 8510 bis zu 10 Inch. Ein weiterer Unterschied ist die Schreibgeschwindigkeit, denn der 1550B schafft 120, der 8510 aber 180 Zeichen pro Sekunde.

Wesentlich vielfältiger als die Unterschiede sind die gemeinsamen Fähigkeiten dieser beiden Druckertypen. Dem Konzept nach wurden die beiden Hauptfunktionen eines Punktmatrixdruckers, nämlich die Textausgabe und die Grafikausgabe optimiert. Das Schriftbild ist bereits in der schnellen Normalschrift sehr klar und die Buchstaben wohlgeformt. Einzelne Punkte sind kaum noch wahrzunehmen, nur an den Kanten der Buchstaben sind kleine Ecken sichtbar. Etwas größer als die Normalschrift ist die Picaschrift, bei der allerdings wieder mehr einzelstehende Punkte erkennbar sind. Ausgeglichen wird dieser Nachteil in der Fettschrift, die besonders deutlich, aber auch merklich langsamer ist. Die Krönung der verfügbaren Schriften aber ist die Proportionalschrift. Sie gleicht nicht nur die Zeichenabstände optimal aus, sondern verwendet auch eine andere Zeichendefinition. Der Probeausdruck (Bild 2) zeigt, daß diese Schrift den Anforderungen der täglichen privaten, aber auch der geschäftlichen Korrespondenz gerecht wird. Der Programmiervorgang ist relativ einfach, denn er wird mit den üblichen ESC-Sequenzen vorgenommen.



Bild 1. Der »kleinere« der beiden Drucker, der Itoh 8510

Matrixdrucker mit Schreibmaschinenqualität

Damit diese Druckfunktionen aber auch richtig aufs Papier kommen, bedarf es der richtigen Verbindung zum Computer. Für die Besitzer des C 64/VC 20 ist dies eine relativ problemlose Angelegenheit, denn beide Drucker haben in der Regel eine Centronics-kompatible Schnittstelle. Die Commodore-Grafik- und Cursorsteuerzeichen sind im Zeichensatz der Itoh-Drucker nicht vorhandenen. Erst die Verwendung eines speziellen Soft- oder Hardwareinterfaces mit entsprechender Programmierung, macht diese Zeichen verfügbar. Die zweite Anschlußmöglichkeit der Itoh Drucker besteht im Einbau einer seriellen Schnittstelle.

Sehr erfreulich verlief der Praxistest mit einem Textverarbeitungspro-

Fortsetzung auf Seite 161.

Die Schriften des Itoh 8510 und 1550B

Mit der Normalschrift schafft der Ihoh 180 Zeichen pro Sekunde

Die Picaschrift wird gerne verwendet

Für Hervorhebungen eignet sich die Fettschrift

Auch die Eliteschrift ist verfügbar.

Am schönsten ist aber die Proportionalschrift, die schon fast Briefqualität hat.

Das Unterstreichen gehört fast schon zum Standard.

Die Schmalschrift sorgt für mehr Zeichen auf einer Seite.

Bild 2. Ein Schriftbild wie es nur von sehr wenigen Matrixdruckern erreicht wird.



Der Petal MA20 - kleiner Name, großer Drucker

Obwohl das Schriftbild der Matrixdrucker ständig besser wird, sind Typenrad-Drucker in diesem Bereich bislang ungeschlagen. Ein auch für den C 64 bestens geeigneter und preisgünstiger Typenraddrucker ist der Petal MA20.



Bild 1. Der Petal MA20 Typenraddrucker



Leider waren Typenraddrucker bislang immer relativ teuer. Nicht so der Petal MA20 (Bild 1), er ist baugleich mit dem bekannten Juki 6100 Typenraddrucker, unterscheidet sich von diesem aber im Preis. Der Juki hat einen empfohlenen Verkaufspreis von zirka 1798 Mark. Der Petal MA 20 soll laut Liste 1498 Mark kosten. Er ist aber unserer Erfahrung nach noch um einiges günstiger zu haben. Der Petal MA20 ist ein Typenraddrucker, das heißt er besitzt im Gegensatz zu einer Schreibmaschine keine eigene Tastatur. Die Texteingabe und Drucksteuerung muß von einem Computer übernommen werden. Mit diesem Konzept sind natürlich Vor- und Nachteile verbunden. Hauptnachteil ist für den Commodore 64-Besitzer, daß keine Grafik- und Commodore-eigene Schriftzeichen ausdruckbar sind. Außerdem sind Typenraddrucker oft relativ langsam und laut (beim Petal zirka 62dBA). Der Vorteil des exzellenten Schriftbildes überwiegt diesen Nachteil aber dann bei weitem, wenn hauptsächlich Texte ausgedruckt werden. Durch Auswechseln der Triumph-Adler-kompatiblen Typenräder (100 Zeichen) steht eine enorme Anzahl der verschiedensten Schriftarten zur Verfügung. Wissenschaftliche Sonderzeichen, Schreibschrift und fremdsprachliche Schriften sind so innerhalb von Sekunden verfügbar. Aber auch das zum Drucker mitgelieferte Typenrad Caroll-Pica erlaubt schon einige Variationen der Schrift (Bild 2). Nicht vergessen wurde dabei die Unterstreichfunktion und die Fettschrift. Der Haupteinsatzbereich eines Druckers wie dem Petal MA20 ist sicherlich die Textverarbeitung. Mit einem guten Textverarbeitungsprogramm, wie dem Vizawrite 64 (das eine eigene Druckoption für den Petal/Juki hat), verwandelt sich die gute Stube in ein kleines Büro. Der Petal MA20 kann sowohl Einzelblatt-, als auch Endlospapier verarbeiten. Für unbeaufsichtigtes Drucken sind zusätzlich ein automatischer Einzelblatteinzug und ein Traktoraufsatz erhältlich. Für den normalen Betrieb genügt aber die Standard-Ausrüstung. Die aber kann sich sehen lassen: Die Druckgeschwindigkeit beträgt 18 Zeichen pro Sekunde bei bidirektionalem Druck. Sehr praktisch ist der sowohl hard- als auch softwaremäßig bestimmbare Zeichenabstand. Er beträgt entweder 10, 12 oder 15 Zeichen pro Zoll. In der Standardschrift können so bis zu 165 Zeichen in einer Zeile untergebracht werden. In der ebenfalls verfügbaren Proportionalschrift (automatischer Ausgleich der Zeichenabstände) sind sogar 220 Zeichen pro Zeile verfügbar. Damit ist auch gesagt, daß die Papierbreite bis zu 13 Inches (= 33 cm) betragen kann. Der Petal MA20 verarbeitet somit auch DIN-A4-Papier im Querformat.

Die unumgängliche Wartezeit auf das fertige Schriftstück wird durch den eingebauten Pufferspeicher von 2 KByte (erweiterbar auf 8 KByte) angenehm verkürzt. Angeschlossen wird der Petal MA20 an den C 64/VC20 wie jeder andere Drucker mit Centronics-kompatibler Schnittstelle. (Arnd Wängler/aa)

Bezugsquelle: Weber Computertechnik, Eulenspiegelstraße 56, 8000 München 83, Tel. 089/6012554

Der Typenrad-Drucker Petal MA20

Der Typenraddrucker Petal MA20 kann mit einem Centronics-Interface direkt an den Commodore 64 angeschlossen werden.

Er beherrscht eine Reihe sehr interessanter Druckfunktionen:

1. Unterstreichen von Texten
2. **Fettschrift für besonders wichtige Dokumente und Hervorhebungen**
3. Dies ist die Proportionalschrift, bei der alle Zeichenabstände ausgeglichen werden.
4. **Die Shadow Schrift eignet sich besonders für Hervorhebungen**
5. Natürlich sind auch Subskript und Superskript mit dem entsprechenden Typenrad einstellbar. Die Typenräder sind Triumph-Adler kompatibel.
6. Befehle zum Einrücken und Zentrieren fehlen natürlich auch nicht.

Bild 2. Einige der möglichen Schriftarten



DRUCKSYMPATHIE

Nicht nur die gute Optik, sondern auch sehr gute Leistungen kennzeichnen den BMC BX100. Dabei ist dieser Matrixdrucker auch noch sehr preisgünstig.



Bild 1. Leistungsfähig und schön — der BMC BX100

Ein guter Matrixdrucker kostete vor etwas mehr als einem Jahr noch ein halbes Vermögen. Das hat sich geändert. Der BMC BX100 (Bild 1) kann als Paradebeispiel dieser Entwicklung bezeichnet werden. Er bietet für relativ wenig Geld (anschlußfertig 1200 Mark) Leistungen, die auch im professionellen Einsatz kaum Wünsche offen lassen.

Schon nach dem ersten Augenschein wirkt er sympathisch, ja man kann sogar von einem hübschen Gerät sprechen. Der Drucker ist kompakt aufgebaut. Das wurde vor allem durch den im Gehäuse versenkten Antriebsmechanismus für Einzelblätter und Endlospapier möglich. Der Vorteil dieses Konstruktionsprinzips liegt in dem auch für die beachtliche Druckgeschwindigkeit von 100 Zeichen pro Sekunde angenehmen Geräuschpegel. Der BMC BX100 wird in der Regel mit einer Centronics-kompatiblen Schnittstelle ausgeliefert. Es sind aber auch eine serielle Schnittstelle und, besonders interessant, eine Version zum direkten Anschluß an den Commodore 64/VC 20 erhältlich. Zum Test stand die Commodore-Version mit externem Interface zur Verfügung. Die ohnehin umfangreichen Druckfunktionen des BX100 werden durch das Commodore-Interface noch um einige Varianten erweitert. Dazu gehört auch der gesamte Commodore-Zeichensatz und die Cursor-Steuerzeichen. Sogar reverse Buchstaben sind enthalten. Die für jeden Programmierer besonders wichtigen Programmlistings werden komplett mit allen Grafikzeichen und den reversen Steuercodes ausgegeben. Der Drucker wird dabei kaum langsamer. Aber auch die Standardfunktionen des BX100 brauchen den Vergleich mit anderen Druckern nicht zu scheuen: Kursiv-, Proportional-, Elite-, Schmal-, und Breitschrift sind mit einfachen ESC-Befehlen zu aktivieren (Bild 2).

Befehle zur Formatierung des Textes, wie sie von jedem guten Textverarbeitungsprogramm vorausgesetzt werden, sind dem BX100 nicht unbekannt. Dazu gehören Funktionen wie das Anspringen bestimmter Tab-Positionen, das Setzen des linken und rechten Randes oder die Festlegung der Seitenlänge. Man kann sogar das Papier rückwärts transportieren. Das Druckbild kann zwar nicht mit einem Typenraddrucker verglichen werden, ist aber mit das Beste, was wir bei einem Matrixdrucker dieser Preisklasse bisher gesehen haben. Verglichen mit einem Epson FX80 ist das Schriftbild des BX100 sogar noch etwas besser. Verantwortlich für diese sehr guten Druckeigenschaften ist der Druckkopf, der mit seiner 9x11 + 3sp Nadelmatrix üppig ausgestattet ist.

Mit seinen zwei verschiedenen Nadeldichten kann der BX100 als voll grafikfähig bezeichnet werden, zumal das verwendete Interface diese Funktionen voll unterstützt. Zum Ausdruck des Textes oder der Grafik beträgt die einstellbare Papierbreite zwischen 4,5 und 10 Inches (11,43 bis 25,40 Zentimeter). Die Handgriffe für die Justage des Traktors sind einfach und als durchaus fingerfreundlich zu bezeichnen. Ebenso das Einlegen der Farbbandkassette. Das Eindrehen eines einzelnen Blattes ist Dank des leichtgängigen Drehknopfes und der sinnvollen Papierführung problemlos.

Mit 100 Zeichen pro Sekunde liegt die Druckgeschwindigkeit des BX100 etwas über dem Durchschnitt dieser Klasse. Der Druckkopf schreibt dabei in beiden Richtungen (bidirektional) mit Druckwegoptimierung. Beim Grafikausdruck wird unidirektional geschrieben. Diese weit über den Standard hinausgehende Druckfunktionen machen den BMC BX100 zur idealen Ergänzung des Commodore 64/VC 20.

Bis auf das zwar ausführliche, aber leider noch englische Handbuch, ist der BMC BX100 in dieser Preisklasse nur schwer zu überbieten.

(Arnd Wängler/aa)

```
        COMMODORE-GRAFIK:

*     REVERSE-SCHRIFT IST AUCH MOEGLICH
*     Gross- und Kleinschreibung
*     Das ist Eliteschrift

*     DAS IST KURSIVSCHRIFT
*     Dies ist Schmalschrift mit dem BX 100
*     KURSIVSCHRIFT UND SCHMALSCHRIFT
*     Schmalschrift und Breitschrift gemischt

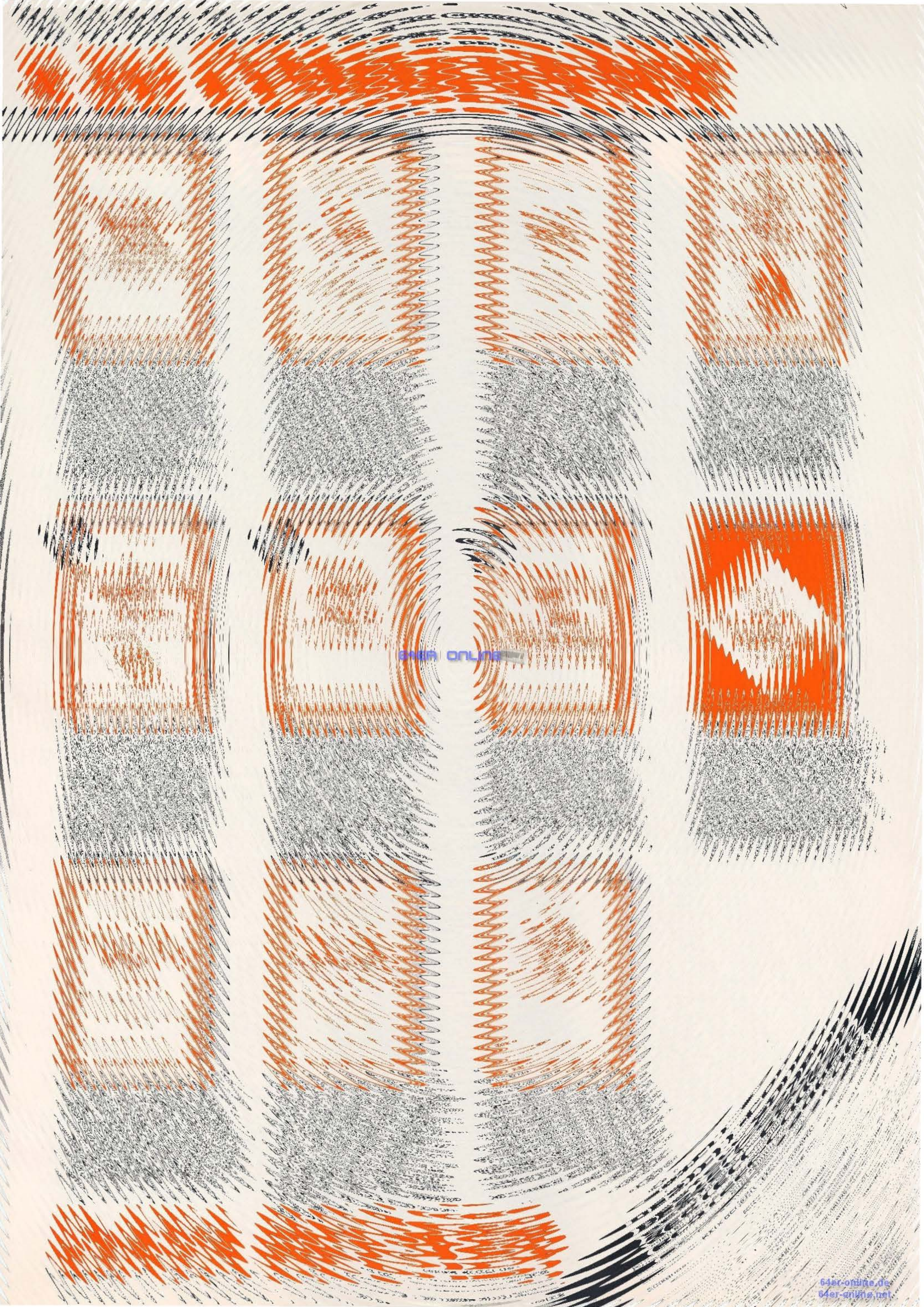
*     Hier ist Breitschrift
*     GRAPHIC:
```

Bild 2. Das sehr gute Schriftbild des BX100

Bezugsquelle: Weber Computertechnik, Eulenspiegelstr. 56, 8000 München 83, Tel. 089/ 6012554




64ER ONLINE



Wie SUPER ist

Supergraphik 64 ist ein Programm, dessen Befehle sich nicht nur auf die Grafik beziehen.

Nun, gehen wir den Befehlssatz von »Supergraphik 64« mal Schritt für Schritt durch. Da wäre als erstes der GMODE-Befehl zu nennen. Er ist die Schaltzentrale der Supergraphik, denn mit ihm bestimmen Sie, was angezeigt, und was befehligt wird. Denn zusätzlich zu den beiden hochauflösenden Seiten kann in der Textseite eine 80 x 50-Blockgrafik erstellt werden, und zwar mit genau denselben Befehlen. Es ist möglich, die Grafikseite 1 anzuzeigen, während Seite 2 oder die Textseite bearbeitet wird. Natürlich wird auch der Multicolormodus unterstützt. Allerdings ist es aus syntaktischen Gründen nicht möglich, eine Seite »normal« anzuzeigen und die andere in Multicolor zu bearbeiten. Mit diesem Befehl kann auch ein Textfenster in der Grafik definiert werden, allerdings nur im Normal-, nicht im Multicolormodus. Zum Textfenster ist zu sagen, daß die obere Kante nicht ganz flimmerfrei ist, was durch die zahlreichen Umschaltungen im VIC bedingt wird.

Hat man erst einmal eine Moduswahl vorgenommen, so kann man mit dem PLOT-Befehl fleißig Punkte setzen und löschen. Hier ergeben sich zusätzlich noch die Möglichkeiten, Punkte zu invertieren, wie auch einen Grafikcursor an die entsprechende Stelle zu setzen. Dieser Grafikcursor kann dann bei weiteren Befehlen als Ausgangspunkt benutzt werden. Ebenfalls mit dem PLOT-Befehl können Linien und sogar Linienzüge gezeichnet werden. Zu den Optionen »Setzen«, »Löschen«, »Invertieren«, »Grafikcursor Bewegen«, treten nun noch »Punktieren« und »Zählen«. Beim »Zählen« wird in eine beliebige Variable die Anzahl der abgefahrenen Punkte übergeben. Auch können zwei Linien, die nirgends im selben 8 x 8-Kästchen verlaufen, verschiedene Farben haben, obwohl Sie sich im »normalen« Modus befinden. Eine Option bewirkt, daß gleichzeitig mit der Linie die entsprechenden Bytes im Farb-RAM gesetzt werden.

Nun gibt es aber nicht nur Linien, auch Kreise wollen gezeichnet sein. Der zuständige Befehl heißt (natürlich) CIRCLE. Aber CIRCLE kann noch mehr. Auch Ellipsen und Vielecke sind kein Problem, es können sogar ein Anfangs- und ein Endwinkel für das Zeichen mitgegeben werden. Natürlich funktioniert CIRCLE auch mit allen schon bei PLOT angegebenen Optionen.

Weitere Befehle sind FRAME, mit dem ein beliebig dicker, rechteckiger Rahmen gemalt werden kann, sowie FILL, das allerdings nur ausgefüllte Rechtecke zeichnen kann. Das Ausfüllen beliebiger umrandeter Flächen ist mit Supergraphik nicht möglich.

Mit TEXT kann ein Text aus einem String an eine beliebige Stelle in eine Grafik hineinkopiert werden. Im 80 x 50 Modus erhalten Sie so vierfach vergrößerte Buchstaben. Natürlich kann die so entstandene Grafik auch wieder gelöscht werden; der Befehl lautet GCLEAR.

Mit dem INVERS-Befehl kann man die beiden Grafikseiten invertieren. Hier ist die Angabe einer Bitmaske möglich, um gestreift zu invertieren. Leider ist es weder möglich, bereichsweise zu invertieren, noch kann die Textseite invertiert werden.

Mit GCOMB lassen sich die beiden Grafikseiten verknüpfen; neben dem einfachen Kopieren einer Seite in die andere kann wahlweise eine UND-, ODER- oder EXKLUSIV-ODER-Verknüpfung durchgeführt werden.

Mit dem TRANS-Befehl wird die Textseite in eine der beiden Grafikseiten kopiert. Daraus ergeben sich völlig neue Dimensionen, wenn Sie einen Sieben-Nadel-Drucker, beispielsweise den MPS 801 besitzen. Dann können Sie nämlich über den HCOPY-Befehl die original 8 x 8-Matrix der Bildschirmzeichen ausdrucken!

In dieser Übersicht der allgemeinen Befehle fehlt nur noch GMOVE. Damit ist ein Verschieben oder Scrollen von Bildschirmzeilen nach links oder rechts möglich. Welche Zeilen verschoben oder gescrollt werden sollen, kann angegeben werden. Allerdings können auch im hochauflösenden Modus nur 8 x 8-Blöcke gescrollt werden.

Kommen wir als nächstes zu den selbstdefinierbaren Figuren. In einem Definitions-String muß angegeben sein, in welcher Richtung der Grafikcursor weiterbewegt, beziehungsweise wann ein Punkt gesetzt werden soll. Die so definierte Figur kann mit dem DRAW-Befehl an jeder Stelle des Bildschirms gezeichnet werden. Auch hier dürfen sämtliche oben angesprochene Optionen verwendet werden. Mit SCALE= können Sie vorher noch übergeben, mit welchem Vergrößerungsfaktor und um welchen Winkel gedreht ihre Figur erscheinen soll. Eine Anwendung sehen Sie in der Bildschirmaufnahme.

Die nächste Gruppe von Befehlen sind die Farbbefehle. Mit COLOR= können, für jeden der drei Bildschirme einzeln, Hintergrund- und Rahmenfarbe gesetzt werden. SCOL= bestimmt die Zeichenfarbe auf der gesamten entsprechenden Seite; sie kann mit PCOL= für einzelne Bildelemente abgeändert werden, um mehrfarbige Grafiken zu erhalten. Im Multicolor-Modus werden mit SCOL= auch die zwei zusätzlichen Farben angewählt.

Dem Speichern und Laden von Grafiken dienen die Befehle GSAVE und GLOAD. Da das Format, in dem ihre Grafiken geladen oder gespeichert werden, frei wählbar ist, können Bilder der verschiedensten Malprogramme geladen und weiterverarbeitet werden. Dummerweise scheint es kein Einheitsformat zu geben, so daß Sie beim Speichern oder Laden immer eine Formatangabe machen müssen, was auf die Dauer ziemlich lästig ist.

Mit HCOPY lassen sich Grafiken zu Papier bringen. Supergraphik unterstützt hierbei die verschiedensten Druckertypen, sogar den Farbdrucker Seikosha GP-700. Um nicht zuviel Speicherplatz für die Hardcopy-Routinen in Anspruch nehmen zu müssen, werden Sie beim Laden

Listing 1. Der DATA-Erzeuger

```
0 REM DATAMAKER-ZUSATZ ZUM SPRITEFORMER
1 REM VON DATA BECKER'S SUPERGRAPHIK 64
2 REM BORIS SCHNEIDER . 64'ER NR. 11/84
3 INPUT"STARTZEILE,FILENAME";SZ,F$
4 OPEN 1,8,2,F$+",R,P": GET#1,A$,A$
5 FORX=1TO21: PRINT SZ+X"DATA ";
6 FORY=1TO3: GET#1,A$: A$=A$+CHR$(0)
7 PRINT RIGHT$("  "+STR$(ASC(A$)),3);
8 PRINT CHR$(-(Y<>3)*ASC(","));
9 NEXTY: PRINT: NEXTX: CLOSE1: PRINT

READY.
```

die Supergrafik?



der Erweiterung nach Ihrem Druckertyp gefragt. Sie haben die Wahl zwischen den Commodore-Druckern 1525, 1526, MPS801, den Seikosha-Druckern GP 100-VC und GP 700A sowie Epson-Druckern mit Data Becker-Interface. Die entsprechende Routine wird dann nachgeladen.

Kommen wir nun zu den Sprites. Hier liegen einige Schwachpunkte. Denn bevor Sie mit den Spritebefehlen arbeiten können, müssen Sie erst einmal ein paar Sprites haben, und zwar in Form von 63 DATAs, wie in den guten alten Zeiten, in denen Sie noch keine Supergraphik hatten. Um diesem Mißstand abzuhelfen, hat Data Becker der Supergraphik-Diskette einen relativ komfortablen Sprite-Editor beigefügt. Dieser hat allerdings drei Mängel: Er mag keine Multicolor-Sprites; er ist aufgrund von eigenen Maschinenroutinen nur dann lauffähig, wenn Supergraphik nicht geladen ist; die von ihm erzeugten Disketten-Files können immer noch nicht von Ihnen und der Supergrafik verwendet werden. Ein kleines abgedrucktes Programm liest angeblich die Files von der Diskette und gibt die entsprechenden Zahlenwerte aus. Aber: Das abgedruckte Programm funktioniert nicht! Deswegen geben wir mit dem Listing 1 allen Supergraphik-Besitzern eine Routine in die Hand, mit der Sie sehr einfach DATA-Zeilen aus SPRITEFORMER-Files erzeugen können. Alles, was Sie jetzt zu tun haben ist: Startzeilennummer und Filename eintippen, 15 Sekunden warten, HOME und 21x RETURN drücken. Haben Sie alle Sprites im Speicher, einfach die Zeilen 0 bis 9 löschen und den Rest auf Disk speichern. Wenn Sie die Sprites später brauchen, einfach mit MERGE nachladen.

Mit SREAD können die 63 Daten einer Sprite-Definition in einen String eingelesen werden. Das geht wesentlich schneller als mit einer FOR-NEXT-Schleife. Mit SDEFINE können Sie dann einem der Sprites eine der Stringdefinitionen zuteilen. Dies hat den Vorteil, daß Sie die Definition für ein einzelnes Sprite schnell wechseln und somit Sprite-interne Bewegungsabläufe auch in Basic programmieren können. Mit SMODE legen Sie die übrigen Eigenschaften eines Sprite wie Multicolor, Farben, Größe und Priorität gegenüber dem Hintergrund fest.

Und wie kriegen Sie das nun auf ihre Mattscheibe? Dazu dient der Befehl SSET. Sie können damit nicht nur Sprites setzen, sondern auch bewegen, indem Sie Start- und Zielkoordinaten sowie die Geschwindigkeit angeben. Das Sprite läuft dann von selbst, und Ihr Programm etwas langsamer, das sich das ganze interruptgesteuert abspielt. Wenn Sie auf das Eintreffen eines Sprites am Ziel warten wollen, so hilft ihnen der SWAIT-Befehl weiter.

Und nun zu den in der Werbung angekündigten 16 Sprites, die gleichzeitig auf dem Schirm erscheinen sollen. Ich halte diese Formulierung für ein wenig übertrieben, 2 mal 8 Sprites wäre wohl richtiger. Denn mit dem SPOWER-Befehl können sie ein Bildschirmfenster definieren, in dem 8 andere Sprites erscheinen können, also auf dem restlichen Bildschirm. Sie haben beispielsweise oben 8 Sprites und unten 8 Sprites.

Aber die Supergraphik bietet nicht nur Grafik-Befehle, auch die Tongeneratoren des SID können per Basic angesteuert werden. Mit VOLUME=, legen Sie erst einmal die Lautstärke fest. SOUND dient der Einstellung von Wellenform und Hüllkurve der drei Stimmen. Mit FILTER kann der Filter des SID gesteuert werden. Und als letztes gibt es TUNE, das nach Voreinstellung der oben genannten Parameter einen Ton spielt.

Supergraphik benötigt eine Menge Speicherplatz, Ihnen gehen 10 KByte Ihres Basic-Speichers und sämtliche RAM-Bereiche in der ROM-Gegend, auch der $C-Bereich verloren. Deshalb wird die Supergraphik wohl kaum mit irgendeiner anderen Basic-Erweiterung zusammenarbeiten können. Aus diesen Gründen wurden die »wichtigsten« Programmierhilfen in das Programm mit eingebaut. So bringt der Befehl DIRECTORY selbiges ohne Programmverlust auf den Bildschirm. Mit MERGE können Pro-

Fortsetzung auf Seite 157



Viel zu schade, um nur damit zu kalkulieren

Multiplan wurde vor Jahren von der amerikanischen Softwarefirma Micro-Soft für 16-Bit-Computer entwickelt und entpuppte sich vom Start weg als der Renner auf dem Software-Markt.

Seit geraumer Zeit nun findet sich Multiplan 64 mit deutscher Benutzerführung und ebensolchem Handbuch im Handel. Jedem, der jetzt befürchtet, daß dies eine stark abgemagerte Version ist, sei gleich der Wind aus den Skeptikersegeln genommen: Mit häufigem, aber den Arbeitsablauf kaum störenden Nachladen einzelner Systemdateien arbeitet Multiplan auf dem C 64 befriedigend schnell und mit voller Leistung.

Doch gerade diese Leistung und hier besonders wiederum ihre Vielseitigkeit ist wohl für viele der Stolperstein bei Multiplan, obwohl gerade diese Fülle an Möglichkeiten bei Multiplan dieses Programm immer noch interessant macht — besonders dann, wenn man mit einem neidischen Auge auf die jüngste Generation von integrierter Software bei den »Großen« schielt.

Denn gerade wer sich ernsthaft mit seinem C 64 beschäftigt, das soll heißen, wer mit ihm seine Briefe oder Manuskripte schreibt und sein Haus oder seine Wohnungen verwaltet, oder wer mit ihm seinen kleinen Betrieb auf Trab bringt, wird sich schon oft über das häufige Programmwechseln-Müssen geärgert haben. Und nicht jeder kauft sich deshalb gleich einen neuen Computer. Und genau dort liegt der Casus-Knacksus in der Anwendung von Multiplan Richtig in den Griff bekommen und mit ein paar Kunstgriffen und Tricks liefert es uns auf dem C 64 fast schon die gleichen Features wie wesentlich aufwendigere und teurere Programme für große PCs. Doch dazu später und ausführlicher mehr. Kurz noch mal Generelles: Multiplan ist ein sogenanntes Spreadsheet. Ein aus 63 Spalten und 255 Zeilen bestehendes Arbeitsblatt. Jede der Spalten kann in ihrer Breite variiert, und sowohl Spalten wie Zeilen, also jedes einzelne Feld kann mit Daten, Formeln, Funktionen oder Texten gefüllt werden. Hat man sich erst einmal so eine Tabelle nach eigenen Wünschen aufgebaut, reagiert Multiplan auf die geringsten Veränderungen und berechnet getreu den Anweisungen alle betreffenden Positionen neu.

Dies setzten wir einmal als mittlerweile hinreichend bekannt voraus und ersparen uns so die trockenen Erklärungen der Wie und Warums und der einzelnen, über 120 Befehle, Funktionen, mathematischen Kürzel und zahlreichen Fehlermeldungen. (Die Nur-Multiplan-Interessierten mögen trotzdem hier noch nicht aufhören weiterzulesen.)

Diese Vielzahl an frei benutzbaren Möglichkeiten macht Multiplan schon fast zu einer eigenen Programmiersprache. Und sie erklärt, warum diese Software so gleichermaßen beliebt und unverstanden geblieben ist.

Doch kommen wir zum Wesentlichen, zum Programm selber. Jeder Neuerwerber wird erstmal positiv überrascht sein. Denn für seine knapp dreihundert Mark, die er für die C 64-Version von Multiplan dem Händler auf die Theke blättern muß, bekommt er genau 1,56 K. Diesmal handelt es sich aber ausnahmsweise nicht um die Meßlatte für Computerleistung oder Softwarelänge, sondern um die guten alten Kilogrämmer der Marktfrauen. Genausoviel wiegt nämlich das in einer Acrylbox verpackte Handbuch samt Systemdiskette.

Vorbildliche Dokumentation

Und zu diesem, selbstverständlich ebenfalls deutschen Handbuch, fallen einem eigentlich nur höchstlöbliche Worte ein. Es ist eines, das diesen Namen wirklich verdient. Microsoft hat hier nicht nur ein Nachschlagwerk geschaffen, sie liefern auch gleich ein komplettes und gut aufgemachtes Lehrbuch mit. Alles in allem zeigt dieses 432 Seiten starke Ringbuch, wie gut und auführlich man Bedienungsanleitungen machen kann. Und man fragt sich, warum es dennoch so viele zusätzliche Bücher, auch noch für jeden Computer ein eigenes, über dieses Programm auf dem Markt gibt.

Doch was bietet nun das Multiplan 64 dem Anwender, außer der Möglichkeit, die Auswirkung einer 5prozentigen Preiserhöhung auf seinen Energie- oder Wirtschaftshaushalt zu kalkulieren? Kann es mehr, als nur ausrechnen, ob nun 6,78%

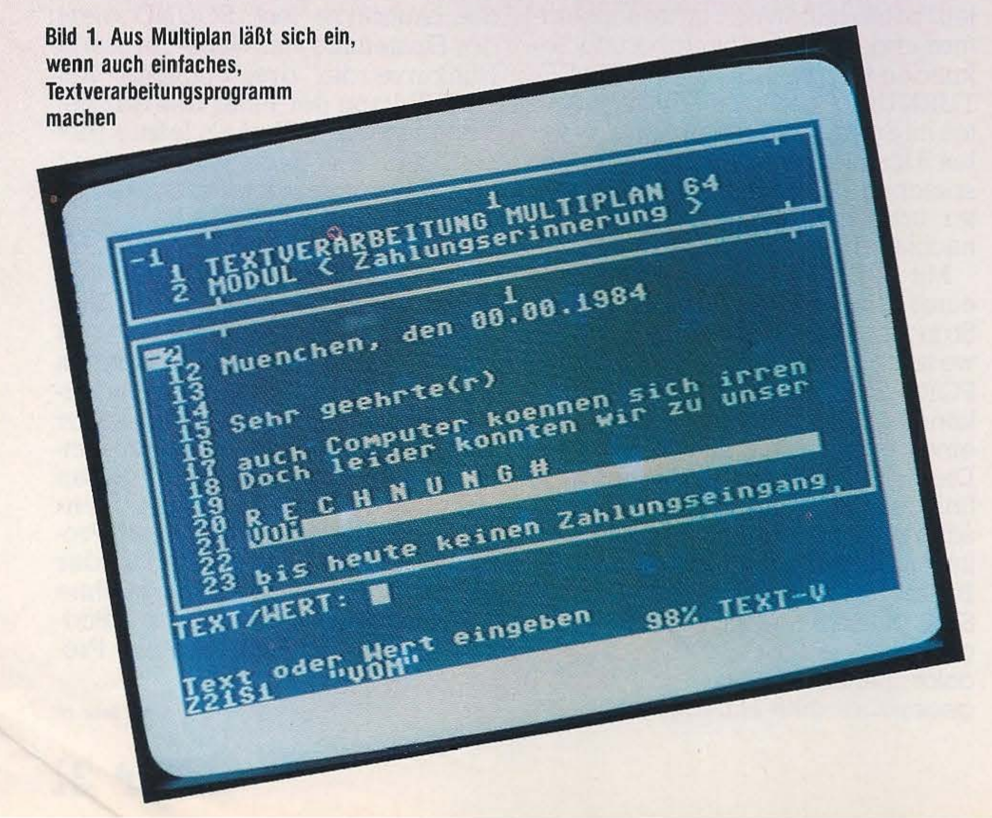


Bild 1. Aus Multiplan läßt sich ein, wenn auch einfaches, Textverarbeitungsprogramm machen

Wer bisher gemeint hat, Multiplan läuft nicht auf dem C 64, erfährt es hier besser. Dieses Tabellenkalkulationsprogramm ist schon seit Jahren für Personal Computer auf dem Markt. Wer aber immer noch glaubt, Multiplan sei nur zum Rechnen und Kalkulieren da, sollte ganz aufmerksam weiterlesen.

Zinsen auf 35 Jahre oder 7,86% Zinsen auf 5 Jahre günstiger sind? Und lohnen sich die drei Blauen für die meistverkaufte Software der Welt in einem Handwerksbetrieb oder Kleinversand, damit der Inhaber dann elektronisch weiß, daß er bei 3,5% mehr Spanne tatsächlich auch mehr verdient? Um es gleich vorwegzunehmen: JA. Man muß nur wissen wie ...

Wer sich durch die Befehlsvielfalt von Multiplan 64 durchgewühlt hat, wird vielleicht schnell feststellen, daß sich dieses Programm ganz ausgezeichnet in einzelne, immer wieder aufrufbare Module unterteilen läßt. Ein etwas größeres für zum Beispiel die Rechnungsstellung oder das Angebotswesen mit wiederum kleineren Modulen für die jeweiligen Posten und Kosten. Durch die Option der Formeln hat man bei der Gelegenheit auch gleich noch einen Überblick über den augenblicklichen Stand der geschriebenen Angebote, Kosten, Einnahmen oder was auch immer gewünscht wird.

Für eine kleine und natürlich nur eingeschränkt komfortable Textverarbeitung reicht es schon aus, die Spaltenbreite entsprechend zu vergrößern und die Felder mit dem Befehl »Zusamm(en)« zu kennzeichnen (Bild 1). Schon schreibt Mulitplan brav den eingetippten Text Zeile für Zeile ins Arbeitsblatt. Der Ausdruck des Geschriebenen erfolgt mit dem Befehl »Z(eilen)/S(palten)-Nummern: Ja (Nein)« auch ohne die, bei Briefen unangebrachten, Numerierungen.

Noch besser geeignet ist Multiplan für das Angebots- und Rechnungswesen (Bild 2). Hier zeigt es seine wahren Stärken, wenn man mal von dem Nur-Kalkulieren absieht. Da Angebots- und Rechnungstexte meist gleichlautend sind, bedarf es hierfür nur je eines Moduls. In diesem sind Text, Artikeldaten und natürlich die Formeln (Artikel mal Stückpreis gleich Angebot plus Mehrwertsteuer) »programmiert«. Bei Anfrage und noch besser nach Lieferung holt man sich das Modul als Datei in den Rechner, braucht nur noch die gelieferten Waren einzugeben und ab die Post.

Auch eine Hausverwaltung ist kein Problem. Einmal programmiert, liefert die Datei die Vorgaben, und die Mieter haben ihre Nebenkostenabrechnung wenig später auf dem Tisch (Bild 3). Den Anwendungen sind praktisch keine Grenzen gesetzt. Der kleine Betrieb spart sich zum einen den permanenten Programmwechsel und zum anderen auch Geld für teure Einzel-Software. In Kauf nehmen muß er dafür nur einige Stunden Anpassungsarbeit und die, allerdings rechnerbedingte, manchmal und bei ziemlich vollem Arbeitsplatz etwas langsame Geschwindigkeit, hat dafür aber eben alles in einem.

Wem diese Arbeit dennoch zuviel ist, kann sein Multiplan 64 auch weiterhin nur für seine Kalkulationen unterbelasten. Was allerdings schade wäre. (Klaus Koch/aa)

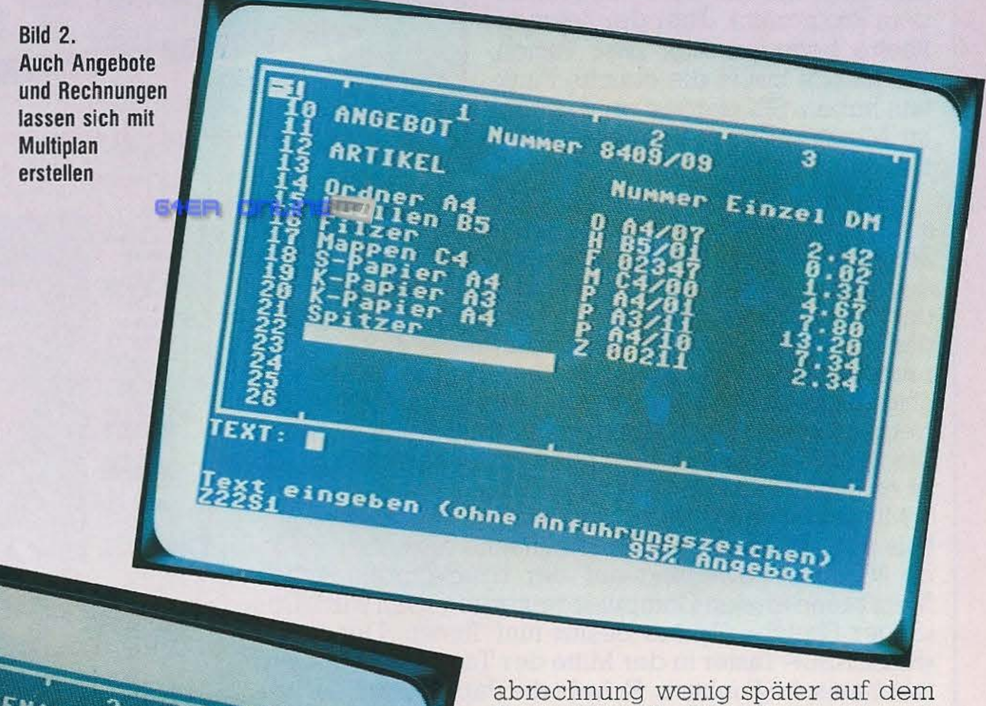


Bild 2. Auch Angebote und Rechnungen lassen sich mit Multiplan erstellen

Bild 3. Eine Abrechnung der Mietnebenkosten — kein Problem

David und



Das Grafik-Tablett »Super Sketch« von PPI (Personal Peripherals, Inc) ist fast vier mal so groß als das »Koala Pad« von Koala Ware. Dem »Koala Pad« liegt eine Diskette mit dem dazugehörigen Programm »Koalapainter« und einigen Beispielbildern bei. Bei PPI wurde das Programm »Graphics-Master« für den »Super Sketch« in ein Steckmodul gepackt. Auf der mitgelieferten Diskette befindet sich lediglich ein Demo-Programm, um die Fähigkeiten des »Graphics-Master« aufzuzeigen. Die Kabel beider Tabletts werden in den Control Port 1 des C 64 gesteckt.

Das »Koala Pad« besitzt eine dunkelgraue druckempfindliche Zeichenfläche, die es erlaubt, mit einen mitgelieferten Stift oder mit dem Fingernagel am Bildschirm zu zeichnen (natürlich nur mit dem dazugehörigem Programm). Über der Zeichenfläche befinden sich zwei Tasten, die jedoch beide die gleiche Funktion haben. Sie werden zur Auswahl im Menü und zum Zeichnen benötigt.

Das Zeichnen mit dem »Super Sketch« ist leider nicht so komfortabel wie mit dem »Koala Pad«. Hier wurde ein Zeiger über einen Kontrollarm mit zwei Paddles verbunden. Diese Version ist in der Herstellung billiger, hat jedoch den großen Nachteil, daß beim Zeichnen immer der ganze Arm mitbewegt werden muß. Dies führt zu recht zackigen Bewegungen. Trotz der größeren Zeichenfläche des »Super Sketch«, werden oft feinere Bewegungen des Kontrollarms nicht wahrgenommen, da die Paddles diese Bewegungen kaum registrieren. So ist exaktes Zeichnen recht schwierig.

Mit dem »Koala Pad« ist das Zeichnen von kleinen Details trotz der kleineren Zeichenfläche einfacher, da jede kleinste Bewegung auf der druckempfindlichen Membrane an den Computer weitergegeben wird. Das »Super Sketch«- Tablett besitzt fünf Tasten. Der große »RELEASE«- Taster in der Mitte der Tastenreihe hat nur mechanische Funktion. Er hebt die Papierhalter auf beiden Seiten des Tabletts, an denen man Vorlagen befestigen kann.

Damit das Tablett sowohl von Links- als auch von Rechtshändern bequem bedient werden kann, wurden jeweils am linken und am rechten Rand der Tastenreihe eine »LIFT«-Taste angebracht. Wenn eine dieser Tasten gedrückt ist, kann der Kontrollarm bewegt werden, ohne daß der »Graphics-Master« am Bildschirm zeichnet. Beim »Koala Pad« ist dies umgekehrt, denn hier muß beim Zeichnen eine der beiden Tasten gedrückt werden. Die »LIFT«-Taste hat weiterhin die Aufgabe, das Menü des »Graphics-Master« zu verlassen und auf Zeichenbetrieb umzuschalten. Weiterhin befinden sich noch eine »MENÜ«-Taste und eine »SELECT«-Taste auf dem Tablett. Mit der »MENÜ«-Taste wird der Zeichenmodus unterbrochen, und am linken Bildschirmrand erscheint das Hauptmenü. Mit dem Kontrollarm kann nun auf die

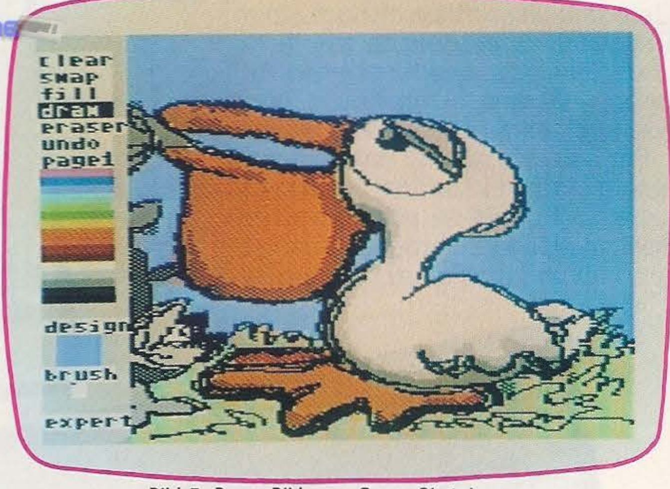


Bild 5. Demo-Bild von »Super Sketch«

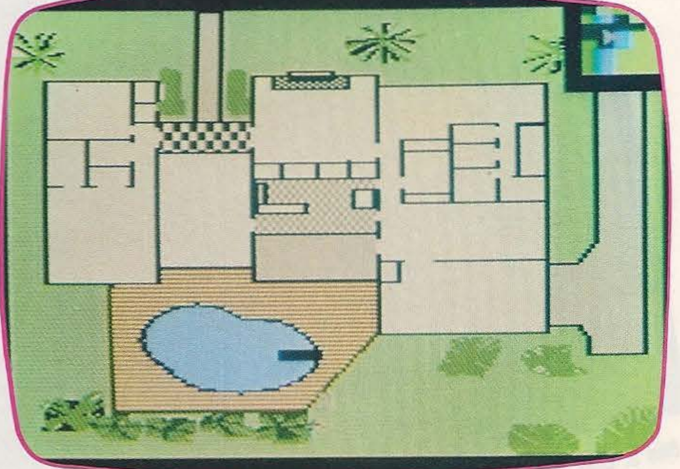


Bild 4. Zoom-Funktion von »Super Sketch«



Für den C 64 sind inzwischen diverse Zeichenprogramme auf dem Markt.

Sie werden mit dem Lichtgriffel, dem Joystick oder mit Paddles gesteuert.

Interessant sind in diesem Zusammenhang auch Grafik-Tabletts.

Goliath

gewünschte Option gezeigt und mit der »SELECT«-Taste ausgewählt werden.

Software im Steckmodul

Beide Programme (Koalapainter und Graphics-Master) besitzen fast die selben Fähigkeiten. Wie schon erwähnt muß der »Koalapainter« im Gegensatz zum »Graphics-Master« erst von der Diskette geladen werden. Nach dem beeindruckendem Titelbild erscheint das sehr übersichtliche Menü. Hier kann zunächst die Pinselform ausgewählt werden. Es stehen acht verschiedene Stärken zur Verfügung. Das Auswählen ist denkbar einfach. Man drückt zunächst mit dem Stift oder dem Finger auf die Oberfläche des Zeichentabletts, und es erscheint ein Cursor, den man nun durch Stift- oder Fingerbewegung auf das gewünschte Symbol führt und eine der beiden Tasten drückt. Genauso kann mit dem Cursor eine Farbe aus der Farbpalette ausgesucht werden. Zum Zeichnen wählt man mit dem Cursor das Symbol »Draw« an, und schaltet auf den Zeichenbildschirm um. Dies geschieht, in dem man mit den Stift an das untere Ende des Zeichentabletts fährt, so daß der Cursor nicht mehr sichtbar ist, und eine der oberen Tasten betätigt. Das Umschalten zum Menü erfolgt auf die gleiche Weise. Zum Zeichnen benötigt man nun beide Hände, da der Pinselstrich erst angenommen wird, sobald eine der beiden Tasten gedrückt ist. Läßt man die Taste los, fährt der Cursor über das Bild, ohne einen Spur zu hinterlassen. Doch »Draw« ist nur eine von 17 verschiedenen Optionen des »Koalapainters«.

So können mit »Frame« beliebig große Vierecke ins Bild gesetzt werden. Mit »Box« füllt man diese gleich mit der gewählten Farbe. Die Option »Circle« generiert Kreise. Mit »Disk« werden diese ebenfalls farbig ausgefüllt. Hier sei erwähnt, daß leider nur die Größe der Kreise und nicht deren Form beeinflußbar ist, so daß diese nicht rund sondern oval auf dem Bildschirm erscheinen. Durch »Xcolor« kann eine Farbe durch eine andere ersetzt werden. Die Option »Mirror« teilt den Bildschirm durch vertikale und horizontale Teilung in vier gleich große Teile. Zeichnet man nun in einem der vier »Quadranten« so wird dies an der X- und Y-Achse gespiegelt und erscheint auch in den angrenzenden »Quadranten«, natürlich spiegelverkehrt. »Line« verbindet zwei Punkte durch eine Linie miteinander. Die Option »Lines« beschränkt dies nicht auf zwei Punkte. Es können also beliebig viele Punkte mit einer Linie verbunden werden. Mit »Rays« läßt sich ein Strahlenbündel erzeugen. Die Option »Fill« füllt eine begrenzte Fläche mit einer bestimmten Farbe auf.

Sehr interessant ist »Zoom«. Mit ihr kann ein bestimmter Bildschirmausschnitt vergrößert und herangeholt werden (Bild 2). Außerdem erscheinen am unteren Rand des »Zoom«-Fensters alle 16 Farben, so daß Details sehr genau gezeichnet werden können. Zu Beachten ist allerdings, daß in einer 8x8-Matrix nicht mehr als vier Farben vorkommen dürfen. »Copy« dupliziert einen beliebigen Bildausschnitt. Hier sind auch Überlagerungen möglich. »Swap« schaltet zwischen ersten und zweiten Zeichenbildschirm um. Hat man einen entscheidenden Fehler gemacht, zum Beispiel eine falsche Fläche ausgefüllt, so kann dies durch den Befehl »Oops« rückgängig gemacht werden. Das Bild erscheint dann wie vor dem letzten Wechsel zum Menü. Mit »Erase« kann die Zeichnung wieder gelöscht werden. Die 17te und letzte Option »Storage« schaltet auf das Disk-Menü um. Hier können Bilder geladen, abgespeichert und Disketten formatiert werden. Auf der Diskette befinden sich einige Kunstwerke wie zum Beispiel das Bild »Jungle« (Bild 1).

Super Sketch

Das Programm »Graphics-Master« befindet sich auf einem Steckmodul. Wird die Menü-Taste gedrückt, erscheint eine leere Zeichenfläche, an deren linken Seite sich das Hauptmenü befindet — leider nicht so übersichtlich wie bei »Koalapainter«. Im Gegensatz zu »Koalapainter« kann hier aber die Pinselform selbst kreiert werden. Hierfür wählt man mit dem Kontrollarm die Option »brush« an. Zum Zeichnen stehen natürlich auch alle 16 Farben zur Verfügung. Hier funktioniert das Zeichnen jedoch anders. Das Zeichenbrett registriert jede Bewegung und hinterläßt, wenn die »LIFT«-Taste nicht gedrückt wurde, ungewollte Spuren in der Zeichnung.

David und Goliath

Bild 2. Zoom-Funktion von »Koala Pad«





Bild 3. Demo-Bild von »Koala Pad:



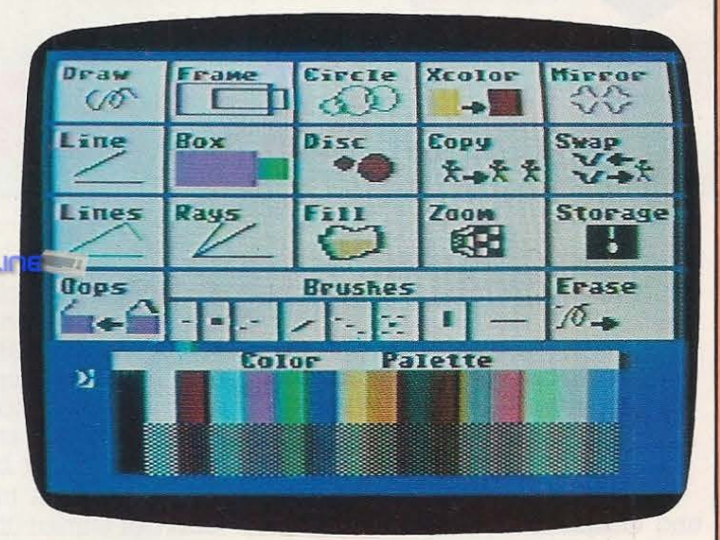


Bild 1. Das ganzseitige Menü von »Koala Pad« mit »Koalapainter«

Nun zu den einzelnen Befehlen: »clear« löscht logischerweise den Bildschirm. Die Befehle »swap«, »fill« und »draw« haben gleiche Funktionen wie bei »Koalapainter«. Die Option »eraser« simuliert einen Radiergummi. Mit »undo« können (wie mit »Oops« beim »Koalapainter«) Fehler rückgängig gemacht werden. Auch der »Graphics-Master« besitzt zwei Zeichenflächen, zwischen denen mit »page« umgeschaltet werden kann. Beim »Koalapainter« werden bei »Fill«, »Box« und »Disk« nur begrenzte Flächen mit Farbe aufgefüllt. Beim »Graphics-Master« bieten sich mit »design« weitere Möglichkeiten, Flächen auszufüllen. So kann ein Zeichenmuster entworfen werden, mit dem eine Fläche aufgefüllt wird.

Mit der letzten Option im Hauptmenü »expert« wird auf das »Expert-Menü« umgeschaltet. Hier finden sich die Befehle »lines«, »rays«, »circle«, »box«, »copy« und »zoom« wieder, die auch dieselbe Bedeutung wie beim »Koalapainter« haben. Hier sei jedoch erwähnt, daß es mit »circle« möglich ist, wirklich runde aber auch ovale Kreise (Ellipsen) darzustellen.

Kommt man mit einem Kreis über den Bildschirmrand hinaus, so wird dieser im Gegensatz zum »Koalapainter« nicht hinausgeschoben sondern »staut« sich am Bildschirmrand. Das »zoom«-Fenster ist leider nicht so komfortabel, da keine Farbänderung vorgenommen werden kann. Das Fenster erscheint am linken unteren Eck der Zeichenfläche und ist nicht annähernd so groß wie beim »Koalapainter« (Bild 4). Mit der Option »h or v« können nur horizontale oder vertikale Linien gezeichnet werden. »window« erlaubt es, daß sich bestimmte Befehle wie »clear« oder »fill« nur auf den durch »window« gekennzeichneten Bereich, beschränken.

Die Option »copy 1/2« kopiert Bildausschnitte von der ersten zur zweiten Zeichenfläche oder umgekehrt. »quad« hat die gleiche Funktion wie »Mirror« beim Koalapainter, während »mirror« beim »Graphics-Master« den Bildschirm nur vertikal trennt. Die Option »flip« spiegelt jede Tätigkeit an einer horizontalen Trennlinie des Bildschirms. Mit »show« können Sie Ihr Meisterwerk bewundern, ohne daß ein flackernder Cursor Sie ablenkt. Die Option »reset« schaltet »window«, »quad«, »mirror«, »flip« und »zoom« wieder aus. »files« hat die selbe Bedeutung wie »Storage« beim Koalapainter. Nur kann hier auch mit Kassette gearbeitet werden. Es bleibt hier noch zu erwähnen, daß mit »Super Sketch« und »Graphics-Master« auch die Bilder vom »Koalapainter« geladen und bearbeitet werden können. Künstlerisch gelungen ist der Pelikan (Bild 5).

Beide Handbücher sind recht ausführlich. Jedoch muß man seine Englischkenntnisse wieder hervorkramen, da beide Bedienungsanleitungen englisch abgefaßt sind. Dem »Super Sketch« liegen außerdem einige Vorlagen bei, die auf dem Tablett befestigt werden können. Beim Nachzeichnen braucht man einiges Geschick, um mit dem Kontrollarm exakt der Vorlage zu folgen.

Da die Hardware von PPI dem »Koala Pad« unterlegen ist, versucht man mit zusätzlichen Funktionen im »Graphics-Master« wieder an Boden zu gewinnen. Ob dies gelingt ist Ansichtssache.

(Christian Quirin Spitzner/rg)

Grafik hoch zwei — das Extended Graphik System

Eine der interessantesten, aber auch schwierigsten Anwendungen des Commodore 64 ist die Grafikprogrammierung. Mit dem Extended Graphik System geht es einfacher.



Die Bearbeitung von Hires-Bildern ist einfach



Mit dem Extended Graphik System hat Interface Age ihre bekannte Ex-(tended-)Produktreihe fortgesetzt und dem Programmierer ein taugliches Werkzeug in die Hand gegeben. Die Konzeption dieses Programmpaketes beruht auf der Philosophie, dem Grafikinteressierten im Handbuch Kenntnisse zu vermitteln, die dieser dann sofort in der Praxis erproben kann. Das umfangreiche deutsche Handbuch hat deshalb auch den Charakter eines gut gemachten Lehrbuches. Angefangen mit der Darstellung des VIC 6567 (Video Interface Chip) über die Erklärung der Speicheraufteilung bis zum Anwendungsbeispiel wird Schritt für Schritt vorgegangen.

Für die praktischen Übungen wird nicht nur das Standard-Basic eingesetzt, sondern das von Diskette ladbare Extended Graphik System. Diese Erweiterung verändert den Speicher des Commodore 64. So werden beispielsweise neue Befehle implementiert, und der Videobereich verlegt. Nach dem Start des Graphik Systems beginnt der Basic-Speicher beispielsweise bei Speicherstelle 1024 (früher lag da der Bildschirm). Insgesamt bleiben dem Anwender 15,5 KByte für seine Programme. Das ist allerdings mehr, als die Zahl andeutet. Weder für einen neuen Zeichensatz noch für bis zu 175 Spritedaten geht etwas vom Basic-Speicher verloren. Weiterer Vorteil des Grafik-Systems ist es, daß der Bereich für Autostartmodule weiterhin verfügbar ist. Das bekannte Extended Basic Level II kann zur gleichen Zeit eingesetzt werden. Das Grafik System ersetzt die im Extended Basic weitgehend fehlenden Grafikbefehle.

Die Leistungsfähigkeit der neuen Grafikbefehle, die übrigens alle mit dem Vorsatz »CALL« aufgerufen werden, zeigen die mitgelieferten Hilfsprogramme. Dazu gehören Zeicheneditor, ein Spritegenerator und ein Grafikeditor. Mit dem Zeicheneditor ist es eine Leichtigkeit den Zeichensatz den persönlichen Wünschen anzupassen. Der Spriteeditor dient dem schnellen Entwerfen von Single- und Multicolor Sprites. Am leistungsfähigsten ist aber der Grafikeditor, mit dem eigene Hires-Grafiken hergestellt, beziehungsweise bestehende verändert werden können. Zum Befehlsvorrat des Grafikeditors gehören Funktionen zum Zeichnen von Kreisen, Linien, Rechtecken sowie die Einblendung von Texten in die hochauflösende Grafik. Der Vorteil dieses Konzeptes ist die Flexibilität, denn eine Anpassung und Erweiterung der Hilfsprogramme ist jederzeit möglich.

Der fortgeschrittene Programmierer wird aber wahrscheinlich mehr am Graphik System interessert sein. Seine Erwartungen werden nicht enttäuscht, denn der Befehlsvorrat ist umfangreich und den gewachsenen Kenntnissen angepaßt. Die Schwerpunkte liegen bei der Farbgebung (Hires und Text), der Spriteverwaltung und -Bewegung (Single- und Multicolor) sowie der Unterstützung hochauflösender Grafik. Bis auf den Befehl zum Ausfüllen bestimmter Flächen (Fill) ist der Befehlsumfang komplett. Für ein flexibles Werkzeug wie das Extended Graphik System ist es fast schon eine Selbstverständlichkeit, daß Bilder von anderen Grafikprogrammen (Micropainter, Koala, Doodle) eingeladen und verarbeitet werden können. Zur Verarbeitung gehören alle Manipulationen am Bild einschließlich der Hardcopys auf einem grafikfähigen Drucker. Besonders sinnvoll ist die Fähigkeit auch Spriteformen auf allen Druckern auszugeben.

Nicht nur der Anfänger wird es schätzen, daß alle wichtigen Programme im umfangreichen Handbuch abgedruckt sind. Sinnvoller wäre es allerdings, diese Programme auf der ohnehin vorhanden Diskette mitzuliefern, zumal die Listings nicht immer fehlerfrei sind. Auf die vielen Beispielbilder für hochauflösende Grafik hätte statt dessen verzichtet werden können. Auch fehlt der Hinweis, daß Programme, die mit dem Graphik System entworfen wurden, auch nur mit diesem wieder gestartet werden können. Ein kleines Ladeprogramm schafft hier Abhilfe, indem das Graphik System jedesmal vor das betreffende Programm geladen wird. Weitergeben darf man diese Programme allerdings nicht, denn damit wäre das Urheberrecht verletzt.

Das Extended Graphik System ist auch für den fortgeschrittenen Programmierer ein sinnvolles Hilfsmittel. Zusammen mit Exbasic Level II ist es zwar nicht ganz billig, aber fast unschlagbar. Besonders lobenswert ist das Handbuch, das schrittweise in die Grafikprogrammierung einweist und noch nach mehrmaligem Lesen bei wichtigen Fragen herangezogen werden kann.

(Arnd Wängler/aa)

Bezugsquelle: Interface Age, Josephsburgstraße 6, 8000 München 80. Preis: 138 Mark.

Ein Stern wird geboren –

Als erstes deutsches Computermagazin testeten wir Vizastar, die neue Datenbank-Tabellenkalkulations- und Geschäftsgrafiksoftware für den C 64.

In unserer letzten Ausgabe haben wir Vizawrite 64 vorgestellt, eines der leistungsfähigsten Textverarbeitungsprogramme für den C 64, wenn nicht sogar das leistungsfähigste überhaupt. Der Commodore 64 stößt damit in die Klasse der kleinen (aber vor allem preiswerten) Personal Computer vor. Zu den Fähigkeiten eines Personal Computer gehört aber mehr, als nur die Textverarbeitung. Drei der meistgebrauchten PC-Anwendungen sind die Datenverwaltung, die Tabellenkalkulation und die Geschäftsgrafik. Oft werden diese Funktionen in einem Programmpaket (zum Beispiel Lotus 1 2 3) kombiniert angeboten. So auch Vizastar 64. Bis auf die Textverarbeitung sind alle anderen Komponenten in Vizastar 64 vereinigt. Obwohl natürlich die Datenübertragung von einer Funktion in die andere (auch in die Textverarbeitung) möglich ist, soll hier zunächst auf die Leistungsmerkmale der einzelnen Funktionen eingegangen werden.

Vorab aber ein paar allgemeine Worte zu Vizastar 64 an sich. Geliefert wird das Programmpaket auf zwei Datenträgern, einem Modul und einer Diskette. Das Modul beinhaltet einen wesentlichen Teil des Steuerprogramms und lädt automatisch die restlichen Programmteile von der Diskette nach. Nach dem Laden kann die Programmdiskette weggelegt werden. Das gesamte Steuerprogramm (reine Maschinensprache) befindet sich im Speicher. Ab da wird nur noch mit Datendisketten gearbeitet. Die von Vizawrite 64 bekannten Einstellmöglichkeiten auf verschiedene Laufwerkskonfigurationen, die Farbgebung und den angeschlossenen Drucker sind auch in Vizastar 64 enthalten. Ein Centronics-kompatibler Drucker kann mit einem einfachen Userport-Kabel direkt angeschlossen werden, die Centronics-Treibersoftware wird immer mitgeladen. Vizastar 64 unterstützt eine große Anzahl von Druckermodellen wie beispielsweise CBM, Epson, Juki und Brother. Zum Lieferumfang gehört ebenfalls ein 90-seitiges ausgezeichnetes deutsches Handbuch. Ein zusätzliches Übungsbuch, ist nach Auskunft des Herstellers, in Kürze erhältlich. Die getestete Vorab-Version verfügte über keinen deutschen Zeichensatz, in der endgültigen Version soll er aber enthalten sein. Auch die Benutzerführung von Vizastar 64 ist noch englisch, die Kommandos sind allerdings einfach und einprägsam.

Die Tabellenkalkulation

Nach dem zirka 115 Sekunden dauernden Laden erscheint das Arbeitsblatt (Bild 1). Es ist wie ein richtiges Stück grafisches Papier aufgebaut. In den oberen drei Zeilen befinden sich die Menü- und Kommandohinweise. Die jeweils geltende Funktion wird revers dargestellt. Durch Druck auf die »Space«-Taste wird von Menüpunkt zu Menüpunkt gesprungen. Die jeweiligen Unterfunktionen erscheinen in Zeile 2 und 3. Ein Arbeitsblatt besteht aus Zeilen (0 bis 999) und aus Spalten (A bis BL). Jede Spalte kann eine beliebige Breite zwischen 3 und 36 Bildschirmspalten haben. Das Arbeitsblatt ist natürlich viel zu groß um als Ganzes auf dem Bildschirm zu erscheinen. Deshalb ist immer nur ein Teil abgebildet. Mit den Cursortasten wird der Arbeitsblattausschnitt über den Bildschirm verschoben. Ein direkter Sprung an eine Adresse ist natürlich auch möglich. Manchmal wird es notwendig, an verschiedenen Punkten des Arbeitsblattes auf einmal zu arbeiten. Vizastar 64 bietet deshalb eine »Window«-Funktion (Bild 2) an, mit der beliebige Teile des Arbeitsblattes auf dem Bildschirm eingeblendet werden (bis zu acht Windows). Die »optisch« darunter liegenden Informationen gehen natürlich nicht verloren, sondern bleiben weiterhin im Berechnungsprozeß eingebunden. Die Schnittpunkte zwischen den Zeilen und Spalten nennt man Zellen (insgesamt 64000). Jede Zelle hat ihre eigene »Adresse«, so heißt beispielsweise die erste Zelle A0, die rechts daneben liegende B0. Die Zellen sind der klein-





der Commodore 64

ste Informationsträger. Hier werden Texte und Zahlen gespeichert. Da aber jede Zelle ihre eigene Adresse hat, sind die verschiedensten Verknüpfungen zwischen den einzelnen Zellen möglich. Das Ergebnis einzelner Berechnungen wird wiederum in vorher definierten Zellen abgelegt, beziehungsweise beeinflußt alle von diesem Ergebnis abhängigen Zellen. Ein Beispiel soll dies verdeutlichen: Nehmen wir an, Sie wollen die jährlichen Kosten Ihres Autos berechnen. Aber Sie wollen auch wissen, was passiert, wenn der Benzinpreis steigt, oder wenn Sie mehr Kilometer fahren. Diesen Sachverhalt nennt man in der Betriebswirtschaft »Entscheidungsfindung«. Sie erhalten Auskünfte über die Auswirkungen einzelner Maßnahmen in Abhänigkeit von vorgegebenen Werten und Bedingungen.

Bild 3 zeigt, wie so ein Kalkulationsblatt aussehen könnte. Feste Werte sind Anschaffung, Reparaturen und Zubehör. Alle anderen Werten werden von Vizastar 64 anhand von Verknüpfungsbedingungen errechnet. Der Wertverlust wurde im Jahr 1982 mit 15% vom Anschaffungspreis angegeben. Der Anschaffungspreis von 1983 ist natürlich der Restwert von 1982. Die Benzinkosten werden in Abhängigkeit von den gefahrenen Kilometern und dem Benzinpreis berechnet. Ändert man nun auf dem Arbeitsblatt eine der unabhängigen Zahlen, zum Beispiel die gefahrenen Kilometer verändern sich sofort die Benzinkosten, und die Gesamtkosten pro Jahr. In Bild 4 sieht man was geschieht, wenn der Benzinpreis auf 2 Mark und die gefahrenen Kilometer auf 15000 ansteigen. Vizastar 64 bietet eine große Anzahl von Verknüpfungsfunktionen (Bild 5), die für optimale Geschwindigkeit sorgen. Damit aber auch die Eingabe der Werte und Formeln einfach zu handhaben ist, stehen dem Benutzer umfangreiche Editierbefehle zur Verfügung. Sie reichen vom Einfügen und Löschen einzelner Zeilen/Spalten über das Sortieren von Reihen bis zum Kopieren und Verschieben ganzer Blöcke.

Ganz besonders leistungsfähige Funktionen sind die sogenannten EXEC-Befehle. Dieser Operationsmodus verwendet den Inhalt des Arbeitsblattes um die Tastatur zu simulieren. Sich wiederholende Befehlssequenzen (ähnlich einem Basic-Unterprogramm) können in Spalten innerhalb des Blattes eingetippt und zu jedem Zeitpunkt wiederholt ausgeführt werden. Zusätzliche EXEC-Befehle erlauben das bedingte Überspringen zu anderen Zellen. Imposantestes Beispiel für die EXEC-Befehle ist das auf der Systemdiskette mitgelieferte Demo-File, das einen kompletten Arbeitsablauf inklusive bewegter Grafik simuliert. Alle diese Funktionen werden mit unglaublicher Geschwindigkeit ausgeführt (man glaubt gar nicht wie schnell der C 64 sein kann). Das Abspeichern einzelner Arbeitsblätter zur späteren Verwendung ist für Vizastar 64 eine Selbstverständlichkeit. Damit sind die Anwendungsmöglichkeiten des Arbeitblattes natürlich bei weitem noch nicht ausgeschöpft. Aber gerade dank der Flexibilität der Arbeitsblattfunktionen ist es für jeden Anwender möglich, sich seine eigene Problemlösung zu kreieren. Vizastar 64 unterstützt ihn dabei mit seiner Leistungsfähigkeit und der sinnvollen Benutzerführung.

Die Datenbank

Der zweite Bestandteil von Vizastar 64 ist die Datenbankfunktion. Die Operationen dieser Datenbank werden aufgerufen, als ob sie Teile des Kalkulationsprogrammes wären — was sie eigentlich auch sind. Vom Kalkulationsprogramm kann, ohne daß etwas nachgeladen werden muß, direkt in die Datenverwaltung und zurückgesprungen werden (inklusive Datentransfer). Das Arbeitsblatt verwandelt sich dabei zur frei definierbaren Karteikarte. Wie eine solche Karteikarte aufge-



Viza Star

teilt aussehen könnte, zeigt Bild 5. Spezielle Zeichenbefehle ermöglichen den Einsatz des gesamten Commodore-Zeichensatzes. Die Eingabemaske kann somit extrem individuell gestaltet werden. Der Anwender ist nicht an eine Karteikarte gebunden, sondern kann einen einzelnen Eintrag über mehrere Karten ausdehnen. Auch ist die spätere Umgestaltung (Einfügen, Löschen) der einzelnen Karteikarten jederzeit möglich. Der grundsätzliche Aufbau der Datenbank ist mit dem Superbase 64 zu vergleichen. Grundsätzlich kann eine Vizastar 64-Datenbank aus bis zu 15 Einzeldateien bestehen. Ein Datensatz (Record) besteht aus bis zu 1000 Zeichen (Superbase 1108). Pro Satz sind 64 verschiedene Felder mit maximal 128 Zeichen pro Feld möglich (Superbase 127 Felder/255 Zeichen). Bei jedem Feld muß ein Indexschlüssel definiert werden, damit jeder einzelne Eintrag schnell aufgerufen werden kann. Ein Feld, das als Index-Schlüssel dient, hat eine maximale Länge von 30 Zeichen (Superbase 30). Da die Daten gemäß dem Schlüssel in alphabethischer oder numerischer Reihenfolge auf der Diskette geordnet sind, ermöglicht dieser Schlüssel beinahe den sofortigen Zugang zu jedem Datensatz innerhalb des Files. Die Zugriffszeit beträgt dabei im Durchschnitt nur 4 Sekunden pro Record. Die Anzahl der Datensätze (Records) findet bei Vizastar 64 ihre einzige Beschränkung in der vorhandenen Disketten-Speicherkapazität. Beim 1541-Laufwerk kann beispielsweise eine Kundenkartei (Name, Adresse, Tel. Ansprechpartner, Datum) über 500 Datensätze enthalten. Sind zwei Laufwerke angeschlossen, oder die SFD 1002, erhöht sich die Anzahl der Datensätze. Hierbei sei angemerkt, daß der Anschluß der SFD 1001 nicht problemlos ist, da das Commodore IEEE-Interface den Einsatz eines zusätzlichen Moduls mit Autostart nicht zuläßt. Der Zugriff auf die einzelnen Daten eines Files wird mit dem ACCESS-Befehl ausgeführt. Dieser Befehl erlaubt es, jeden Eintrag nach Wünschen einzugeben, abzurufen zu ändern oder zu löschen. Die Auswahlkriterien können definiert werden, so daß nur die gewünschten Einträge ausgewählt werden.

Mit etwas Übung und Geschick dauert es mit Vizastar 64 nicht lange, eine Buchhaltung oder eine Lagerbestandskartei aufzubauen. Im privaten Sektor ist an Anwendungen wie eine Bücher- oder Adreßdatei zu denken. Alle Daten, auch die des Arbeitsblattes, können mit Vizastar 64 auch ausgedruckt werden. Dazu dienen zwei Funktionen. Wenn ein grafikfähiger Drucker angeschlossen ist genügen zwei Tasten um eine entweder 1:1 oder 1:2 Hardcopy auf dem Drucker auszugeben. Nicht grafikfähige Drucker tun das zwar auch, haben aber keine reversen Zeichen beim Ausdruck. Datensätze können in beliebiger Form, zum Bei-

```
Cell  Sheet  File  Print  Data  Graph
Access, Transfer, Use, Setup, Other
      A        B         C
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

Bild 1. Das Arbeitsblatt im Ausschnitt

```
Cell  Sheet  File  Print  Data  Graph
 Copy, Move, Insert, Delete, Title,
  Window, Global, Sort, Erase, Xec
#1       A          B         C
 0Gewinn und Verlustrechnung
 1              -1983|1984-
 2Absatz HW #2      B         C
 3Absatz SW  3
 4Zubehoer   4      #3        C
 5Gesammt    5       5
 6           6       6       #4
 7Einkaufe HW 7      7        7
 8Einkaeufe S 8      8        8
 9Einkaeufe Z 9      9        9
10Gesammt   10      10       10
11          11      11       11
12Ertrag v.St 12    12       12
13Ertrag n.St 13    13       13
14Gewinn    14      14       14
```

Bild 2. Die echte Window-Technik

```
Cell  Sheet  File  Print  Data  Graph
Format, Calc, Protect, Width, Skipto,
         Display, Tone
#1       A               B          C
 0*KFZ-Kosten*          1982       1983
 1Anschaffung          17534    14903.9
 2Versicherung        968.24    919.828
 3Benzinkosten          1935       1320
 4Wertverlust         2630.1    1490.39
 5Reparaturen        1625.78    1354.87
 6Zubehoer               300        300
 7Gesammt           2499.312   2254.332
 8Restwert           14903.9   13413.51
 9
10
11gef.Kilometer        15000      10000
12Benzinpreis           1.29       1.32
13
14
```

Bild 3. Das Arbeitsblatt als Autokosten-Tabelle

Cell Sheet File Print Data Graph
Format, Calc, Protect, Width, Skipto,
Display, Tone

#1	A	B	C
0	*KFZ-Kosten*	1982	1983
1	Anschaffung	17534	14903.9
2	Versicherung	968.24	919.828
3	Benzinkosten	3275	3000
4	Wertverlust	2630.1	1490.39
5	Reparaturen	1625.78	1354.87
6	Zubehoer	300	300
7	Gesammt	2633.312	2440.999
8	Restwert	14903.9	13413.51
9			
10			
11	gef.Kilometer	25000	15000
12	Benzinpreis	1.31	2
13			
14			

Bild 4. So ändern sich die Werte, wenn die Benzinpreise steigen

Format Insert Delete Paint High
Save Quit
General, Integer, Currency, Date, Sci

Adressen	
Firma	<Stahlschmidt OHG>
Name	<Franz Gruber>
Strasse	<Isarring 34>
Wohnort	<Muenchen 2>
PLZNr.	<8000>
Telefon	<089/236732>
TELEX	<8861169 sta d>
Anrede	<Herr>

Letztes Treffen: 10.Oktober83

Ergebnis: Abschl. Auftrag Nr.1423/83

Liquiditaet/Limit:gut/bis 10000,-DM

Bild 5. Ein mögliches Karteiblatt

spiel als Adreßaufkleber auf fast jedem Drucker oder Schreibmaschine zu Papier gebracht werden.

Der dritte Bestandteil von Vizastar 64 ist die Geschäftsgrafik. Der Anwender hat hier die Wahl zwischen Balken- oder Liniendiagrammen (Bild 6 und 7). Die abzubildenden Werte lassen sich aus beliebigen Bereichen des Arbeitsblattes entnehmen. Mit Hilfe der EXEC-Funktion sind sogar Änderungen bestimmter Werte im Zeitverlauf darstellbar. Die Trendanalyse (ein wichtiges betriebswirtschaftliches Instrument) wird somit ungemein erleichtert. Durch die oben besprochene Window-Technik ist auch eine optisch ansprechende Abbildung bestimmter Werte denkbar. Grafik, Zahlen und Kommentare können beliebig gemischt werden.

Fazit: Es gibt keinen Zweifel, Vizastar 64 ist eines der leistungsfähigsten Programme, — aus dem Bereich Datenbank und Tabellenkalkulation —, das bisher in unserer Redaktion getestet wurde. Es zeichnet sich nun auch bei den Heimcomputern — wie bereits bei den PCs seit einem Jahr — der Trend zur integrierten Software ab. Die genaue Beschreibung aller Befehle von Vizastar 64 würde den Rahmen bei weitem sprengen. Zusammen mit Vizawrite 64 ist Vizastar 64 die optimale Programmausstattung für den Commodore 64. Nicht nur der Privatmann, sondern gerade der kleine Unternehmer wird viele seiner täglichen Entscheidungen mit Vizastar 64 leichter und genauer beurteilen können. Der einzige Nachteil dürfte die geringe Kapazität von zirka 9 KByte für Daten sein.

Wie eine Commodore 64-»PC«-Konfiguration aussehen könnte, haben wir in Bild 8 darzustellen versucht. Es besteht aus dem C 64, einem Floppy-Laufwerk, einem Farbmonitor, einem Matrix- und Typenraddrucker sowie aus dem Viza-Paket. Der Preis für das gesamte System liegt bei zirka 6500 Mark (mit den beiden Druckern), wobei Vizawrite 64 und Vizastar 64 mit etwa 700 Mark beteiligt sind. Man könnte in Anbetracht der Leistungsfähigkeit schon fast von »billig« reden.

(Arnd Wängler/aa)

Deutschland: Interface Age, Josephsburgstr. 6, 8000 München 80
Bezugsquellen: Microton, Postfach 40, CH-2542 Pieterlen, Preis: 398,—

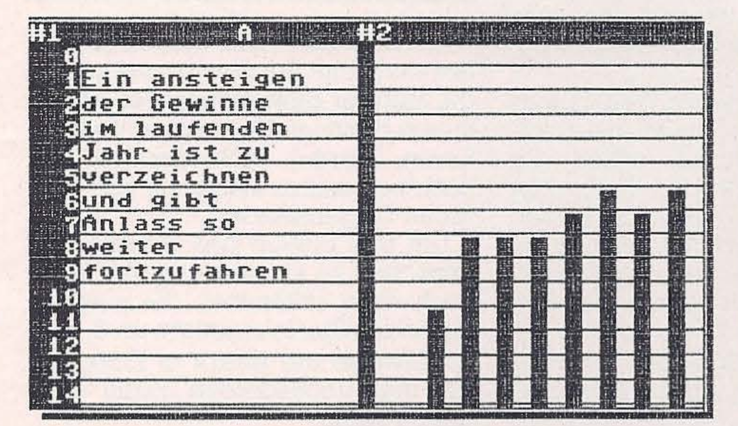
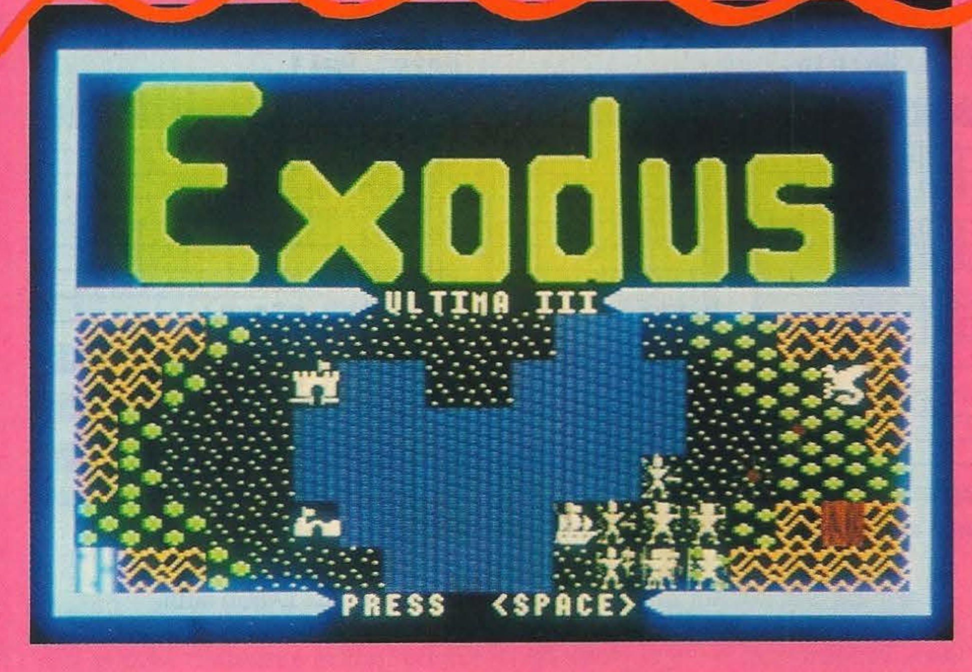


Bild 6. Text und Balkendiagramme



Bild 7. Text und Liniendiagramme

EXODUS – ULTIMA III



Exodus, auch als Ultima III beka zum vorhergehenden Verkaufsh ein Spiel, an dem man entwede zweifelt: Zum Lösen des Spiels Wochen — ein Tagespensum v

Ultima III ist zwar die Fortsetzung zu Ultima II, das man aber zum Spielen von Exodus nicht benötigt. Um die Handlung von Ultima III zu begreifen, muß jedoch auf die Geschichte des zweiten Teils zurückgegriffen werden:

Das Ziel des C-64-Spiels Ultima II war, den bösen »Minax« zu eliminieren. Als dieser schließlich starb (nachdem der Spieler Ultima II gelöst hat), herrschte im Lande schließlich 20 Jahre lang Frieden. Doch die Horden des Bösen kamen wieder. Exodus, der Sohn des toten Minax, wählte den richtigen Augenblick, um das Land »Sosaria« erneut mit Tod und Verderben heimzusuchen. Ihre Aufgabe ist es nun, mit einer Gruppe von vier Abenteurern, das Land Sosaria vom Bösen zu befreien.

Sie sehen sich der Aufgabe gestellt, das Land zu erkunden und das Böse wo es auftritt zu zerstören. Das Endziel ist, Exodus zu finden und zu vernichten.

Zunächst wählt man unter seinen Charakteren vier »Adventurer« aus, mit denen man in eine Welt der Gefahren aufbricht. Zur Wahl stehen zehn verschiedene Charakterklassen (»Fighter, Paladin, Ranger, Barbar, Alchemist, Druid, Cleric, Wizard, Illusionist, Thief«), die man mit fünf verschiedenen Rassen kombinieren kann (»Human, Elf, Dwarf, Fuzzy, Bobbit«). Jede dieser Charakterklassen hat bestimmte Attributwerte: Ein Kämpfer hat seine Stärke, ein Zauberer seine Intelligenz, ein Dieb seine Geschicklichkeit, etc. Die Rasse ist mitentscheidend für die Maximalverteilung der Werte: Ein Zwerg kann seine Stärke auf bis zu 95 Punkte bringen, seine Geschicklichkeit jedoch höchstens auf 50. Die Maximalwerte für Menschen sind für alle Werte 75 Punkte. Zusätzlich steht Ihnen noch die Wahl des Geschlechts Ihrer Spielfiguren offen: Männlich, weiblich oder »other« (was das ist, dürfen Sie sich selbst überlegen).

Durch die geeignete Zusammenstellung Ihrer Charaktere und deren Attributwerte, sichern Sie sich also Ihr Überleben und den eventuellen Erfolg. Bei unglücklicher Auswahl der Charaktere, zum Beispiel lauter schwache Zauberer und nichts anderes, werden Sie sehr bald auch große Mißerfolge erleben müssen.

Wenn Sie in die weite Welt hinausziehen, begegnen Ihnen viele verschiedene Kreaturen, einige gut und viele böse. Sind die Kreaturen Ihnen freundlich gesinnt, so haben Sie die Möglichkeit, mit ihnen zu reden. Dabei können Sie oftmals nützliche Hinweise zur Lösung des Rätsels erfahren. Im Falle, daß Sie auf feindliche Monster stoßen, steht Ihnen meistens ein schwerer Kampf bevor.

Sie sind in diesem Kampf nicht nur auf Nahkampfwaffen oder Weitschußwaffen (zum Beispiel Pfeil und Bogen) beschränkt: Wenn Sie Ihre Spielcharaktere sorgfältig ausgewählt haben, können Ihre »Wizards« oder »Clerics« den Tod der Feinde mit machttollen Zaubersprüchen auch ein bißchen beschleunigen.

Im Verlauf des Spiels treffen Sie auf Städte, Dungeons (unterirdische Labyrinthe, in Exodus acht Stockwerke tief), Burgen und sogar ein »längst vergessenes Land«. Zusätzlich gibt es noch versteckte Städte, die man nur bei genau der richtigen Stellung des Mondes und durch Gehen in den richtigen Teleporter erreicht.

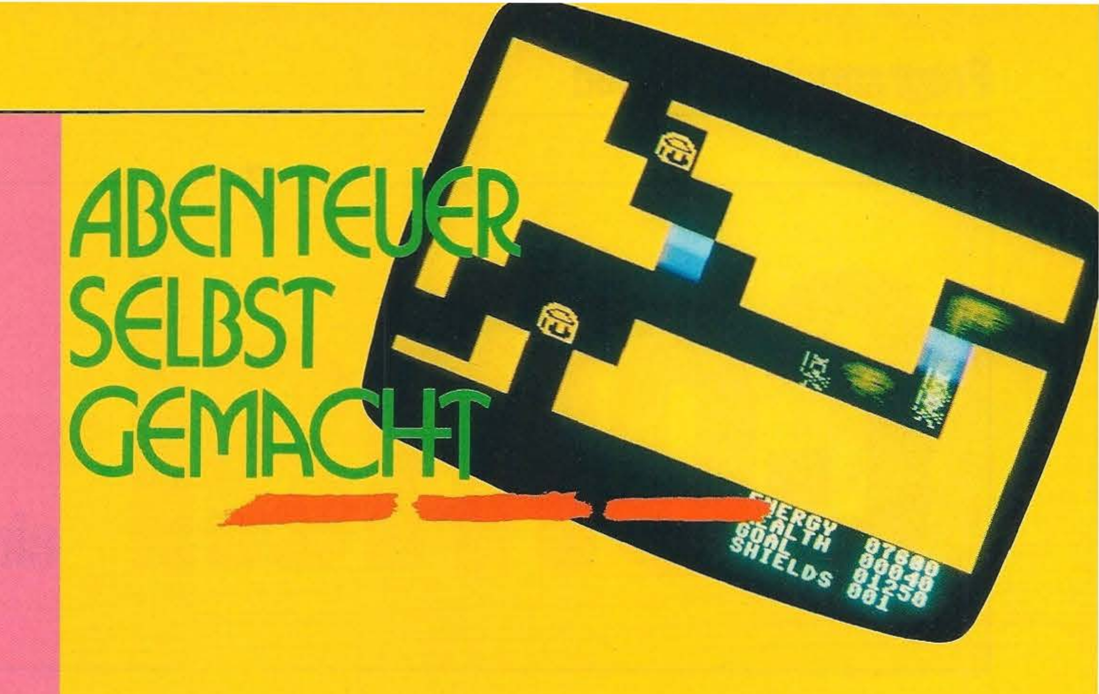
Geduld ist nötig, aber es lohnt sich

In den Städten und Burgen gibt es fast sämtliche Ausrüstung zu kaufen, die man zur Bewältigung des Abenteuers benötigt (Rüstungen, Waffen, Essen, Pferde). Außerdem gibt es noch besondere magische und exotische Gegenstände.

Um den Schluß des Abenteuers erreichen zu können, benötigt man noch viele Hilfsmittel, die man im Laufe des Spiels ergattern muß, beispielsweise die »Mark of Fire«, mit der man ungeschoren durch Lava gehen kann.

Wer Ultima III lösen will, der benötigt viel Zeit und Geduld. Wer bei-

ABENTEUER SELBST GEMACHT



nnt, ist die Fortsetzung it »Ultima II«. Exodus ist r ewig spielt oder ver- benötigt man zirka fünf on 10 Stunden vorausgesetzt.

des aufbringen kann, für den lohnt sich dieses Spiel, auch bei einem Preis von zirka 170 Mark.

Außerdem sollte man beachten, daß im Preis nicht nur die Spielediskette enthalten ist, sondern auch eine luxuriös aufgemachte Verpackung, in der sich einige Besonderheiten befinden: Eine auf Stoff gedruckte farbige Landkarte, Zauberspruchbücher für »Clerics« und »Wizards«, ein Heft mit der Geschichte der Entstehung des Landes und Informationen über Fauna und Flora (Monstercharakteristiken, Zeichnungen und Erläuterungen, Beschreibung der Städte etc.), und natürlich eine genaue Spielanleitung.

Exodus ist ein durchaus empfehlenswertes Spiel, das den Spieler auch über längere Zeit hinweg beschäftigen kann (außer er verzweifelt daran) und nicht so schnell langweilig wird.

(Manfred Kohlen/F.Wlodarczyk/aa)

PS: Wir werden auch weiterhin über interessante Fantasy- und Abenteuerspiele berichten, sowie Lösungshinweise zu besonders schwierigen Abenteuern bieten. Im Augenblick arbeiten wir an einer Lösung zu Encharter, die wir voraussichtlich in der Februar-Ausgabe veröffentlichen werden.Weitere Lösungsvorschläge zu besonders guten und besonders schwierigen Abenteuerspielen sind jederzeit willkommen.

Der »Adventure Creator« von Spinnaker macht es möglich: Der Abenteuer-Spieler kann in selbstprogrammierte Fallen tappen. Oder Freunde ganz schön ins Schwitzen bringen.

Den Begriff »Abenteuer-Spiel« beziehungsweise »Adventure« sollte man in Zusammenhang mit diesem Programm nicht falsch verstehen: Hier ist kein Spiel gemeint, bei dem man Kommandos wie »Go north« oder »Open door« eingibt. Bei »Adventure Creator« handelt es sich um einen sogenannten Abenteuer-spiel-Generator, bei dem man sich verschiedene Szenen selbst aufbauen muß, durch die man sich dann mit Hilfe seines Joysticks durchkämpft.

Ganz Faule können den Marsch durch die Gänge und Zimmer natürlich auch dem Zufall überlassen. Der Computer bestimmt dann, auf welche Tücken der Spieler trifft. Das Programm wurde jedoch eigentlich dazu entwickelt, um sich eigene Abenteuerszenen aufzubauen. In diese Abenteuer kann man die verschiedensten Gegenstände einbauen: Geheimtüren, Fallen, Schatztruhen, heiße Wände, . . .

Von Schildern über Fackeln bis hin zu sogenannten »Hobbles« und »Nippers« kann man alles auflesen und mitschleppen.

In einem richtigen Abenteuerspiel sind natürlich auch die Feinde und Freunde nicht weit: Im Kreaturen-Laboratorium (frei übersetzt nach der englischen Anleitung) entwickelt man die Persönlichkeit und Eigenschaften der Monster. Zur Wahl stehen Kreaturen, die einem etwas geben oder etwas haben wollen, handeltreibende und als Teleporter fungierende, sowie Angreifer.

Bei Angriffen fließt übrigens überhaupt kein Blut. Das einzige, was passieren kann, ist der Verlust von Energie, aber auch Schlaffheit kann zum (friedlichen) Tod führen.

Wenn auf der Packung »ab 8 Jahren« steht, so ist dies durchaus richtig. Spinnaker Software ist bekannt dafür, Lernspiele für Kinder auf den Markt zu bringen. Das soll natürlich nicht heißen, daß Erwachsene nicht genauso viel Spaß daran haben können. Das Spiel fördert die Kreativität des Kindes, indem es ihm erlaubt, eigene Szenen des Adventures auf dem Bildschirm aufzubauen. Der Schwierigkeitsgrad des Adventures ist in einem Maße angelegt, das es auch Kindern ermöglicht, das Abenteuer erfolgreich durchzustehen. Allerdings bedeutet die natürlich, daß der geringe Umfang der Abenteuer Erwachsene schnell langweilt. Die Kinderfreundlichkeit des Programms geht also auf Kosten der älteren Computerfans.

Hier hätte Spinnaker ruhig mehr Fantasie aufbringen können, so daß auch Erwachsene nicht so schnell das Interesse verlieren. Das Spiel kostet schließlich 130 Mark. Eltern von acht- bis zwölfjährigen Kindern kann man dieses Spiel durchaus empfehlen — sofern sie die (einfache) englische Anleitung verstehen und ihren Kindern das Spiel erklären.

(M. Kohlen/F. Wlodarczyk/aa)

Comal – eine Einführung

Ist Ihnen Basic zu wenig leistungsfähig, Pascal zu ermüdend, Assembler zu primitiv und Forth zu merkwürdig? Dann wird es Zeit für Comal.

Teil 1

Die Programmiersprache Comal ist keine ganz neue Sprache, wenngleich sie auch – ähnlich wie Forth – erst seit relativ kurzer Zeit in Europa kursiert. Comal (COMmon Algorithmic Language) wurde bereits 1973 von Borge R. Christensen und Benedict Loefstedt in Dänemark entwickelt, fand aber, wie viele andere Programmiersprachen, zunächst kein großes Echo. Ähnlich wie bei Forth bildeten sich aber bald in vielen Ländern nationale User-Groups, die sich für die Verbreitung und kontinuierliche Verbesserung der Sprache einsetzten.

Das im 64'er Magazin, Ausgabe 8/84, besprochene Comal ist die Version 0.14, das auch als Grundlage für diesen Einführungskurs dient. Comal 2.0 ist um einiges schneller und ist ein 52 KByte langes Programm. Dabei sind mehr als 30 KByte frei für eigene Programme. Das Modul selbst enthält 64 KByte ROM. In Dänemark wird es für umgerechnet 600 Mark angeboten und ist dort fast an allen Schulen im Einsatz. Ältere Comal-Versionen als 0.14 (zum Beispiel 0.12) verfügen unter Umständen über eine Reihe von Befehlen nicht. Umgekehrt ist die Version 0.14 gegenüber der kommerziellen Version 2.0 mit einigen Nachteilen behaftet, zu denen vor allem der mit exakt 9902 Bytes nicht gerade üppig bemessene Speicherplatz gehört. Die Comal-Versionen 0.xx sind sogenannte »Public Domain Software«, das heißt sie dürfen kopiert und weitergegeben, allerdings nicht kommerziell vertrieben werden. Der Sinn dieser auf den ersten Blick verblüffenden Freizügigkeit liegt in der gewünschten möglichst umfassenden Verbreitung der Sprache.

In letzter Zeit gewinnt Comal zunehmende Bedeutung als Alternative zur vergleichsweise primitiven Basic-Programmierung. Der Benutzer von Comal hat dabei gegenüber der Verwendung anderer Programmiersprachen den entscheidenden Vorteil, daß die Sprache stark an Basic angelehnt ist, daß aber die vielen Schwachpunkte von Basic überwunden wurden. Hinsichtlich des Konzeptes der strukturierten Programmierung wurden viele Anleihen bei Pascal gemacht, ohne jedoch dessen oft unnötig komplizierte und langatmige Sprachstruktur zu kopieren.

Doch damit noch nicht genug. Jetzt wird es für den Besitzer eines C 64 erst richtig interessant (für den VC 20 ist Comal leider nicht erhältlich). Comal hat nämlich noch bei einer dritten Programmiersprache Anleihen gemacht, und zwar bei Logo. Damit stehen eine ganze Anzahl von sehr stark an Logo angelehnten Befehlen für hochauflösende Grafik zur Verfügung. Der große Vorteil dieser Befehle ist ihre einfache und übersichtliche Struktur. Es sind kaum Koordinatenangaben nötig, sondern es wird mit einer sogenannten »Turtle« gearbeitet.

Turtle ist die englische Bezeichnung für »Schildkröte«. Dieser etwas ungewöhnliche Name stammt aus Logo, das ursprünglich als grafische Programmiersprache für Kinder entwickelt wurde. Die »Schildkröte« ist dabei ein Grafik-Cursor (bei Comal durch ein Dreieck dargestellt), der auf den gerade aktuellen Punkt im Grafik-Bildschirm zeigt. Die Turtle kann nun mit verschiedenen einfachen Befehlen bewegt werden. FORWARD n bewegt die Turtle um n Grafikpunkte vorwärts, BACK n bewegt sie rückwärts. LEFT w und RIGHT w drehen die Turtle um den Winkel w (in Grad) nach links oder rechts. HOME setzt die Turtle wieder auf ihre Ausgangsposition in der Bildschirmmitte.

Es existieren natürlich noch eine ganze Reihe weiterer Grafikbefehle und sogar etliche Befehle zur Spritesteuerung. Das kleine Demo-Programm (siehe Listing) zeichnet einige Quadrate mit zunehmender Seitenlänge auf den Bildschirm. An diesem Programm sind schon einige grundsätzliche Eigenschaften von Comal erkennbar, zum Beispiel die Programmstrukturierung und das Arbeiten mit Prozeduren. Wenn Sie Comal bereits zur Verfügung haben, dann geben Sie doch dieses kleine Programm einfach einmal ein und lassen sich überraschen.

Wir wollen uns an dieser Stelle jedoch noch nicht weiter mit Einzelhei-

ten wie Prozeduren oder Funktionen beschäftigen, sondern uns dieses sehr umfassende Thema für später aufheben. Auch Comal-Grafik und Sprites werden wir ausführlich in einer der nächsten Folgen behandeln. Wir wollen stattdessen ganz am Anfang beginnen und uns etwas genauer damit beschäftigen, wie man denn nun Comal dazu bekommt, die grundlegenden Dinge wie Programm editieren, laden, speichern und drucken für uns zu erledigen.

Das Arbeiten mit Comal

Kommen Sie mit Basic zurecht? Dann wird Ihnen auch das Arbeiten mit Comal keine Schwierigkeiten bereiten. Genau wie Basic ist auch Comal eine interaktive Sprache. Die meisten Befehle lassen sich auch im Direktmodus ausführen. Soll jedoch eine Programmzeile gespeichert werden, so setzt man einfach — wie von Basic bekannt — eine Zeilennummer davor.

Doch Vorsicht! Einige Unterschiede zu Basic gibt es schon beim Editieren eines Programms. Zum Beispiel dürfen in einer Comal-Zeile nicht mehrere Befehle stehen (der Doppelpunkt fungiert in Comal nicht als Trennzeichen zwischen zwei Befehlen, sondern hat verschiedene andere Funktionen). Auch ist es zum Löschen einer Programmzeile nicht ausreichend, nur die Zeilennummer einzugeben. Dafür gibt es den »DEL«-Befehl, mit dem man nicht nur einzelne Zeilen, sondern auch ganze Zeilenbereiche löschen kann. Die Syntax ist die gleiche wie bei »LIST«. »DEL 50-150« löscht also beispielsweise die Zeilen 50 bis 150. Der »LIST«-Befehl ist in Comal übrigens sehr komfortabel. Das Listen eines Programms läßt sich nämlich durch Drücken der Space-Taste anhalten. Ein zweiter Tastendruck, und das Listen wird fortgesetzt.

Zu beachten ist auch, daß Zeilennummern in Comal maximal vierstellig sein dürfen, sonst wird ein Syntax-Fehler angezeigt. Auch die Null ist als Zeilennummer nicht erlaubt.

Bei der Programmeingabe wird man sehr schnell eine wesentliche Eigenschaft von Comal kennen- und schätzenlernen, nämlich den sofortigen Syntax-Check. Alle fehlerhaften Programmzeilen werden bereits bei der Eingabe mit einer entsprechenden Fehlermeldung zurückgewiesen. Der Cursor blinkt dabei genau an der Stelle, an welcher der Fehler aufgetreten ist.

Interpreter oder Compiler?

Comal führt die meisten Befehle im Direktmodus aus, die Programme werden einfach mit »RUN« gestartet — wie bei einem typischen Interpreter. Andererseits müssen Strings dimensioniert und Variablen vor dem ersten Aufruf einen definierten Wert besitzen — wie bei einem typischen Compiler.

In Wirklichkeit ist Comal keins von beiden — oder beides zur Hälfte, je nach Standpunkt. Das Handbüch spricht von einem »Three Pass Interpreter«, was möglicherweise der Wahrheit am nächsten kommt. Jedenfalls arbeitet Comal tatsächlich in drei Phasen. Die erste Phase kennen Sie bereits, wenn Sie schon Programmierversuche in Comal hinter sich haben. Es ist der Syntax-Check, der unmittelbar nach Eingabe einer Zeile ausgeführt wird. In dieser Phase wird die Programmzeile — ähnlich wie bei Basic — in eine kompakte Form umgewandelt, indem die Schlüsselwörter in Ein-Byte-Abkürzungen, sogenannte Token, umgewandelt werden.

Der Syntax-Check funktioniert im Prinzip recht einfach. Bei jeder eingegebenen Zeile wird zunächst überprüft, ob die Zeilennummer im erlaubten Bereich von 1 bis 9999 liegt. Anschließend wird getestet, ob eine Kommentarzeile vorliegt. Ein Kommentar wird in Comal allerdings nicht mit »REM« eingeleitet, sondern mit zwei Schrägstrichen. Jede Zeile, die mit »//« beginnt, wird daher nicht weiter beachtet. Alle anderen Zeilen durchlaufen jedoch die Routine »Text in Token wandeln«. Dabei wird ganz einfach überprüft, ob das erste Zeichen in der so codierten Zeile ein Token ist. Ein Token erkennt das Comal-System daran, daß in dem betreffenden Byte Bit 7 gesetzt ist; das funktioniert also völlig analog zu Basic.

Ist das erste Zeichen einer Zeile also weder das Kommentarsymbol noch ein Token, dann wird ein Syntax-Fehler gemeldet. Des weiteren wird einfach die Anzahl der öffnenden und schließenden Klammern einer Zeile gezählt. Ergibt sich eine Ungleichheit, dann resultiert das in einer Fehlermeldung. Eine kontextabhängige Syntaxprüfung findet jedoch nicht statt, das heißt, daß die Eingabezeile in dieser ersten Phase ohne Bezug zum Rest des Programms überprüft wird.

Sie können das Prinzip leicht selbst testen, indem Sie beispielsweise die folgende Zeile eintippen:

```
100 print sqr(2)
```

Wie zu erwarten, wird die Zeile widerspruchslos angenommen. Ändern Sie die Zeile jetzt doch einmal in

```
100 print xyz(2)
```

So funktioniert der Syntax-Check

Auch diese Zeile wird widerspruchslos angenommen, da sie mit einem zulässigen Schlüsselwort (print) beginnt und die gleiche Anzahl öffnende wie schließende Klammern enthält. Warum erfolgt hier keine Fehlermeldung ? Es gibt doch gar keine Funktion »xyz(2)«. Wirklich nicht? Was wäre, wenn Sie vor dem Eintippen der Zeile kein »NEW« gegeben hätten, um ein eventuell vorher vorhandenes Programm zu löschen? Woher wollten Sie dann mit Bestimmtheit sagen können, daß es eine Funktion xyz nicht gibt? Außerdem handelt es sich möglicherweise gar nicht um eine Funktion, sondern um ein eindimensionales Feld. Sie merken schon, solange Sie nur diese eine Zeile kennen und vom eventuell vorhandenen übrigen Programm keine Ahnung haben, sind Sie in der gleichen Situation wie der Comal-Interpreter. Der behandelt nämlich bei einer Eingabe auch nur diese eine Zeile und beachtet den Rest des Programms nicht weiter. Daher gibt er in so einem Fall auch lieber keine Fehlermeldung aus, denn möglicherweise wurde vorher im Programm eine Funktion xyz definiert oder ein Feld xyz dimensioniert. Die eingegebene Zeile ist also in Wahrheit tatsächlich syntaktisch korrekt, denn auf eine Print-Anweisung kann ein beliebiger Ausdruck folgen, es muß nicht unbedingt eine der Standardfunktionen oder eine Konstante sein.

Die Überprüfung, ob alle Programmzeilen zusammen auch tatsächlich ein vernünftiges Programm bilden, erfolgt in einer zweiten Pha-



se. Diese Phase wird mit dem Befehl »RUN« gestartet. Im Gegensatz zu Basic beginnt Comal nämlich nach »RUN« nicht unmittelbar mit dem Abarbeiten des Programms, sondern führt zunächst eine Überprüfung der Programmstruktur durch. Dabei wird zum Beispiel festgestellt, ob Schleifen richtig geschachtelt sind und beendet werden, ob jede aufgerufene Funktion und Prozedur auch tatsächlich definiert wurde und so fort. Bei der Gelegenheit werden gleich alle Sprungadressen berechnet und in eine parallel zum Programmtext angelegte Tabelle eingetragen.

Der Begriff »Sprungadresse« ist hier im weiteren Sinn zu verstehen, denn obwohl Comal über ein GOTO-Statement verfügt, sollte dieses im Sinne einer übersichtlichen Programmstruktur nur in Ausnahmefällen verwendet werden. Mit Sprungadressen sind hier also auch »interne« Sprünge gemeint, zum Beispiel auf den Anfang einer Prozedur, wenn diese aufgerufen wird. Damit braucht der Comal-Interpreter beim späteren eigentlichen Programmlauf beispielsweise nicht mehr den gesamten Programmtext nach der Definition einer Prozedur oder Funktion zu durchsuchen, sondern findet die entsprechende Adresse viel schneller durch »Nachschlagen« in einer Tabelle.

Fehlerhafte Programme werden gar nicht ausgeführt

Dieser Vorgang der Ersetzung von symbolischen Adressen (Prozedur-, Funktions-, Variablen- und Labelnamen) durch die tatsächlichen Adressen dieser Objekte im Speicher ist eine der wesentlichen Aufgaben eines Compilers. Insofern hat Comal also tatsächlich Compiler-Eigenschaften. Allerdings werden zum Beispiel arithmetische Ausdrücke nicht in eine andere, maschinennahe Form übersetzt, wie das bei einem »richtigen« Compiler der Fall wäre. Auch findet keinerlei Übersetzung in Maschinensprache statt, so daß es tatsächlich falsch wäre, von einem Compiler zu sprechen.

Nachdem alle Überprüfungen und Übersetzungen abgeschlossen sind, beginnt schließlich die dritte und letzte Phase der Interpretation, nämlich der eigentliche Programmlauf. Diese Phase wird automatisch eingeleitet, wenn in Phase zwei kein Fehler aufgetreten ist. Für den Benutzer sind die Phasen zwei und drei daher in der Regel nicht zu unterscheiden. Daß es sie aber wirklich gibt, kann man mit einem kleinen Testprogramm sehr einfach feststellen:

```
10 print "hier ist Zeile 10"
20 //
30 // es folgt ein Fehler
40 // in der Programmstruktur
50 //
60 endif
```

In Zeile 60 steht das Schlüsselwort für das Ende eines If-Blocks, aber nirgends vorher taucht ein »if ... then« auf.

Ein reiner Interpreter — wie Basic — würde nach »RUN« die Meldung »hier ist Zeile 10« auf den Bildschirm schreiben und erst anschließend feststellen, daß hier ein endif ohne if vorliegt. Lassen Sie aber dieses kleine Programm einmal in Comal laufen und sehen Sie selbst, was passiert. Nach »RUN« wird die Print-Anweisung in Zeile 10 nicht ausgeführt, sondern erst einmal Phase zwei gestartet und das Programm überprüft. Dabei wird der Fehler gefunden, und es erscheint eine entsprechende Meldung. Phase drei, nämlich die eigentliche Programmausführung, wird wegen dieses Fehlers gar nicht erst erreicht.

Phase drei wird überhaupt nur gestartet, wenn ein Hauptprogramm vorhanden ist. Besteht der gesamte Programmtext nur aus Funktions- oder Prozedurdefinitionen, dann wird nach »RUN« ebenfalls nichts ausgeführt. Allerdings können nun — und das ist ganz wesentlich — alle im Programm definierten Funktionen und Prozeduren im Direktmodus aufgerufen werden.

Das ist eine Eigenschaft von Comal, die für den Benutzer von großer Wichtigkeit ist. Denn damit kann, wie sonst in dieser Form nur bei Forth, Logo oder Lisp möglich, der Sprachumfang fast beliebig erwei-

Comal-Befehl	Bedeutung
//	Kommentarzeile
AUTO	automatische Zeilennumerierung
AUTO 100,5	numeriert ab 100 in Fünferschritten
BASIC	Rückkehr ins Basic
CAT	Directory listen, ohne Programmzerstörung
CHAIN "name"	Laden und Starten von Programmen
DEL	Zeilen löschen
DEL 10-30	löscht Zeilen 10 bis 30
DELETE "name"	löscht ein File auf der Disk
EDIT	listet Programm ohne Zeileneinrückung
EDIT 10-30	editieren der Zeilen 10 bis 30
ENTER "name"	sequentielles Programmfile laden
LIST	listet Programm strukturiert
LIST 10-30	listet Zeilen 10 bis 30
LIST "name"	speichert Programm als sequentielles File
LOAD "name"	lädt Programm von Diskette
NEW	löscht Arbeitsspeicher und Variable
PASS "string"	sendet einen Kommandostring an die Floppy
RENUM	Programm in Zehnerschritten neu numerieren
RENUM 100,5	numeriert ab Zeile 100 in Fünferschritten
RUN	startet Programmlauf
SAVE "name"	Programm abspeichern
SELECT	Ausgabegerät wählen
SELECT "LP:"	Ausgabegerät Drucker
SELECT "DS:"	Ausgabegerät Bildschirm
STATUS$	enthält Status des Disk-Kanals
STATUS	Abkürzung für PRINT STATUS$

Tabelle 1. Wichtige Comal-Kommandos und Editierbefehle

tert werden. Benötigen Sie zum Beispiel eine Funktion, ähnlich wie PEEK, die aber nicht nur ein Byte, sondern ein 16-Bit-Wort aus dem Speicher liest? Kein Problem. Geben Sie im Direktmodus NEW ein und danach die folgenden Programmzeilen (Sie können sich das Tippen der Zeilennummern sparen, wenn Sie zuvor den Befehl »AUTO« eingeben):

```
10 func deek(x)
20 wert:=peek(x)+256*peek(x+1)
30 return(wert)
40 endfunc deek
```

Bei diesem Vierzeiler handelt es sich um die Definition einer Funktion »DEEK« mit einem Parameter. In Zeile 20 werden die nötigen Berechnungen ausgeführt. Das Schlüsselwort »RETURN« in Zeile 30 hat eine andere Bedeutung als in Basic. Es besagt, daß die Funktion als Ergebnis des Funktionsaufrufes den Wert der Variablen »WERT« zurückliefern soll. Mit »ENDFUNC DEEK« schließlich wird dem Comal-Interpreter das Ende der Funktionsdefinition angezeigt. Nach »RUN« erfolgt sofort die Meldung »END AT 0040«, und der Comal-Sprachschatz ist um die Funktion »DEEK« erweitert. »PRINT DEEK(209)« zeigt beispielsweise die Speicheradresse des Beginns der aktuellen Bildschirmzeile an.

Doch damit zunächst einmal genug über die Arbeitsweise des Comal-Interpreters. Wenden wir uns nun einigen wichtigen Kommandos zu, die das Arbeiten mit Diskette, Kassette und Drucker ermöglichen.

Wie kommt das Programm auf die Diskette?

Nehmen wir einmal an, wir hätten gerade unser erstes kleines Testprogramm in Comal geschrieben. Natürlich wollen wir unser Erstlingswerk gerne der Nachwelt erhalten. Doch wie bekommen wir das Programm auf die Diskette (oder auch auf die Kassette)? Etwa mit SAVE, wie in Basic? Ja, genauso.

Wie bereits zu Anfang erwähnt, ist Comal stark an Basic angelehnt (siehe Tabelle 1). Die Befehle »LOAD« und »SAVE« dienen zum Laden und Speichern von Programmen. »VERIFY« steht leider in der Comal Version 0.14 noch nicht zur Verfügung. Dafür gibt es aber zusätzlich den »CHAIN«-Befehl, der ein Programm von Diskette lädt und automatisch startet. Im Unterschied zu Basic ist Comal allerdings diskettenorientiert, das heißt Sie können sich das lästige »,8« sparen. Soll allerdings statt auf Diskette auf ein Kassettenlaufwerk zugegriffen werden, so muß die Sekundäradresse 1 angegeben werden.

Zur einfacheren Bedienung der Floppy ist in Comal der Befehl »PASS« vorgesehen, der einen Kommandostring an die Floppy sendet. Soll zum Beispiel eine Diskette neu initialisiert werden, dann braucht nicht erst umständlich der Kommandokanal geöffnet werden, sondern es reicht der Befehl »PASS "I" «. Die Systemvariable »STATUS$« enthält immer den Fehlerstatus der Floppy, und zwar im Klartext. Wem das Eintippen von »PRINT STATUS$« noch zu mühselig ist, der kann auch nur »STATUS« eintippen. Comal versteht dann schon, was gemeint ist.

Erheblich komfortabler als in Basic ist auch das Laden des Directory. Einfach »CAT« (für catalog) eingeben, und das Inhaltsverzeichnis der Diskette wird angezeigt, und zwar ohne das Programm zu zerstören.

Neben »LOAD« und »SAVE« gibt es noch zwei weitere Lade- und Speicherbefehle. »LIST "name" « legt ein Comal-Programm als sequentielles File auf Diskette ab. Mit »ENTER "name" « wird ein solches sequentielles File wieder als Comal-Programm eingelesen.

Was ist der Unterschied zwischen diesen beiden Formen des Abspeicherns? Die Antwort darauf mag der eine oder andere schon von Basic her kennen. Der Comal-Befehl »LIST "name" « hat die gleiche Wirkung wie die Basic-Befehlsfolge »OPEN 1,8,1, "name,s,w" « : CMD 1 : LIST : CLOSE 1«. Anschließend existiert in beiden Fällen auf der Diskette ein sequentielles File, das das entsprechende Basic- oder Comal-Programm als reine ASCII-Zeichenfolge, also ohne Token, enthält.

Das kann durchaus nützlich sein, weil man ein solcherart gespeichertes Programm sehr leicht und ohne Kenntnis der speziellen Befehlstoken weiter bearbeiten kann. Auf der Comal-Diskette befindet sich zum Beispiel ein Demo-Programm »FORMATTER.COM«, mit dessen Hilfe sich solche sequentiellen Files formatiert auf dem Drucker listen lassen, wobei alle wichtigen Parameter eingestellt werden können.

Zum Laden mit »ENTER« gibt es kein einfaches Basic-Äquivalent,

```
0010 // Turtle-Grafik Demo (ev140984)
0020 //
0030 //
0040 // Quadratseite zeichnen
0050 //
0060 proc seite(laenge)
0070  forward laenge
0080  left 90
0090 endproc seite
0100 //
0110 // Quadrat zeichnen
0120 //
0130 proc quadrat(laenge)
0140  for i:=1 to 4 do
0150   seite(laenge)
0160  endfor i
0170 endproc quadrat
0180 //
0190 // === Hauptprogramm ===
0200 //
0210 setgraphic 0 // Hires ein
0220 clear  // Hires loeschen
0230 for laenge:=5 to 50 step 5 do
0240  turtlesize 5
0250  quadrat(laenge)
0260 endfor laenge
0270 end
```

Eine kleine Demonstration der Turtle-Grafik

sondern dort muß man schon ein kleines Programm schreiben. In Comal dient der »ENTER«-Befehl dazu, ein sequentielles File zu lesen und dabei wieder in »normalen« Comal-Text mit Token-Codierung umzuwandeln.

Natürlich wäre es einigermaßen lästig, wenn man Programmlistings nur auf solchen verschlungenen Pfaden auf dem Drucker ausgeben könnte. Aber Comal wäre nicht Comal, wenn nicht alles etwas einfacher ginge als in anderen Sprachen. Da ein Umschalten zwischen Bildschirm- und Druckerausgabe oft gebraucht wird, gibt es in Comal einen speziellen Befehl dafür: Mit »SELECT OUTPUT "gerät" « kann das gewünschte Ausgabegerät gewählt werden. Die Angabe des Schlüsselwortes »OUTPUT« ist dabei nicht unbedingt notwendig. Für »gerät« gibt es zwei Möglichkeiten: »LP:« (Line Printer) für Druckerausgabe, »DS:« (Data Screen) für Ausgabe auf den Bildschirm.

Um also ein Listing auf den Drucker auszugeben, gibt man nacheinander die beiden folgenden Befehle ein:

```
SELECT "LP:"
LIST
```

Nach dem LISTen wird übrigens automatisch wieder auf Bildschirmausgabe umgeschaltet.

Damit sind alle wichtigen Befehle behandelt worden, um Comal-Programme vernünftig editieren zu können. In der nächsten Folge werden wir uns (endlich) dem eigentlichen Programmieren in Comal zuwenden. (ev)

turtle-grafik

Turtle-Grafik ist eine Spracherweiterung, die es in sich hat. Vollständig in Maschinensprache geschrieben stellt sie einige Befehle zur Verfügung, mit denen Sie komfortabel sehr schnell Bilder erzeugen können.

Die schnelle Schildkröte

Wenn Sie dieses Programm abtippen, werden Sie keinen Ärger mit den vielen DATAs bekommen. Haben Sie einen falschen Wert eingegeben, weist Sie die eingebaute Prüfsummenroutine auf die fehlerhafte Zeile hin und listet sie am Bildschirm. Bitte halten Sie sich daher an die Zeilennumerierung. Nach der Eingabe sollten Sie das Programm speichern und erst dann starten. Wenn alles ok ist, empfehle ich Ihnen, das Demoprogramm (Listing 2) einzugeben. Es vermittelt Ihnen einen sehr guten Eindruck von den Fähigkeiten der Grafikerweiterung. Doch nun zur Turtle-Grafik selbst.

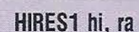
Das Programm ermöglicht die Programmierung von hochauflösender Grafik in Basic mit neuen leistungsfähigen Befehlen.

Wird das Programm gestartet, meldet es sich mit:
TURTLE GRAPHICS
BY PETER MENKE
38911 BASIC BYTES FREE

Nun sind alle Funktionen und Befehle des Programms fest in Basic eingebettet und bleiben bis zum Ausschalten erhalten.

Das Programm unterstützt die Programmierung zweier voneinander völlig unabhängiger Bildschirme:
1. Den normalen Textbildschirm
2. Den Grafikbildschirm auf dem die hochauflösende Grafik erscheint.

Zwischen den beiden Bildschirmen können Sie mit der Funktionstaste F1 hin- und herschalten. Der neue Befehl HIRES1 schaltet den Grafik-Bildschirm ein (HIRES 0 = ausschalten).



Im folgenden die Befehle im Einzelnen

HIRES1 hi, ra
hi = Hintergrundfarbe (0-15)
ra = Randfarbe (0-15)
Die Angaben für Hintergrund und/oder Randfarbe sind nicht unbedingt notwendig, die alten Farben werden dann beibehalten.

CLEAR
Dieser Befehl löscht den gesamten Grafikbildschirm.

REVERS
Dieser Befehl invertiert den gesamten Grafikbildschirm.

COLOR pu, hi, ra
Die Farben der hochauflösenden Grafik werden neu definiert (pu = Punktfarbe). Wie beim HIRES-Befehl können auch hier die Angaben für hi und/oder ra entfallen.

GSAVE »Name«, Gerätenummer
Speichert eine erzeugte Grafik ab.

GLOAD »Name«, Gerätenummer
Lädt eine vorher gespeicherte Grafik in den Computer.

Wenn Sie den Grafikbildschirm einschalten, sehen Sie in der Mitte des Bildschirms einen blinkenden Punkt: Den Grafikcursor (oder »Turtle«, zu deutsch »Schildkröte«).

Durch Bewegen dieses Grafikcursors können Sie Linien auf dem Bildschirm zeichnen.

DEG wi
Bestimmt die Bewegungsrichtung der Turtle. Acht Richtungen sind möglich (»wi« kann Werte zwischen 0 und 7 annehmen).

```
    2
  3   1
 4  *  0
  5   7
    6
```

MOVE x
Move bewegt die Schildkröte um x Punkte.

Der HIRES-Befehl positioniert den Grafikcursor automatisch auf die Bildschirmmitte. Der eingestellte Winkel ist 0. Außerdem wird Modus 0 eingeschaltet (siehe MODE).

LTURN x (x max.=255)
Dreht den Grafikcursor um x Einheiten nach links.

RTURN x (x max.=255)
Dreht den Grafikcursor um x Einheiten nach rechts.

PLOT x-cor, y-cor
Setzt den Grafikcursor auf eine bestimmte Bildschirmposition. Die obere linke Ecke des Bildschirms hat die Koordinaten 0,0; die rechte untere 319,199.

MODE m (m max 4)
Bei den Befehlen MOVE und PLOT kennt das Programm 4 Modi.
0 = Punkt setzen
1 = Punkt löschen
2 = Punkt invertieren
3 = nichts verändern
Normalerweise ist Modus 0 eingeschaltet. Mit dem MODE-Befehl läßt sich dies ändern.

Gleichzeitige Darstellung von Text und Grafik

Das Programm teilt den Bildschirm in einen Text- und in einen Grafikteil.

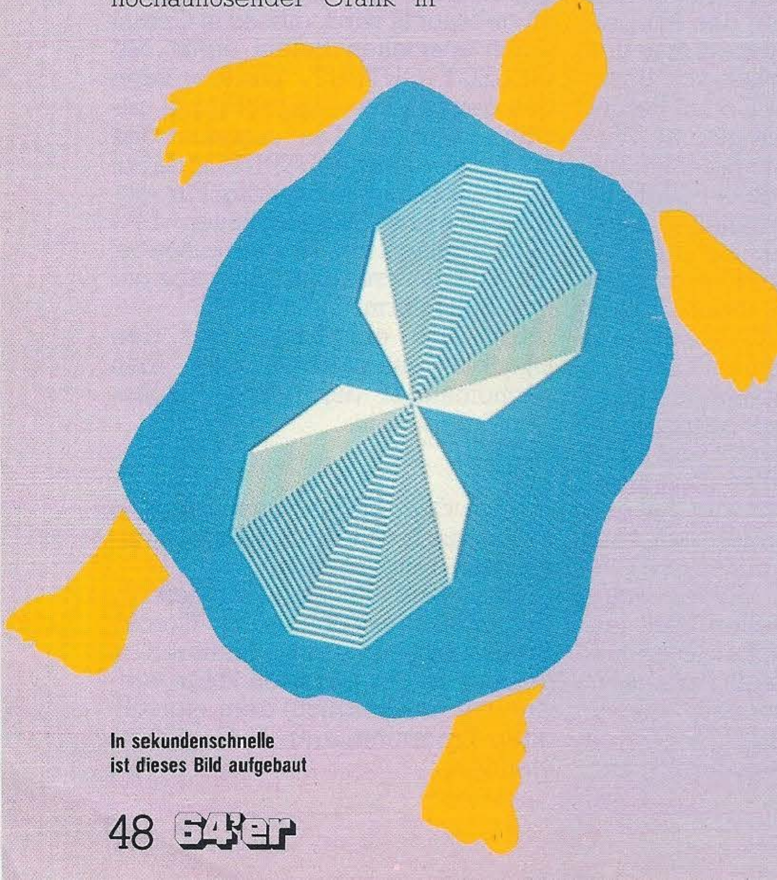
Betätigt man bei eingeschaltetem Grafikbildschirm die F3-Taste, so wird im unteren Teil des Bildschirms der untere Teil des normalen Textbildschirms eingeblendet.

Bei nochmaligem Betätigen dieser Taste wird das »Textfenster« wieder ausgeblendet.

WINDOW 1
Schaltet Textfenster ein.

WINDOW 0
Schaltet Textfenster aus.

Mit der Taste F5 wird der Cursor (der normale) in die obere linke Ecke des Textfensters gebracht. Dies entspricht der HOME-Taste für den gesamten Bildschirm. Die F5-Taste läßt sich ebenso programmieren wie die HOME-Taste, das heißt mit PRINT "(F5)" (auf dem Bildschirm erscheint ein reverses Grafikzeichen) läßt sich der Cursor in die obere linke Ecke des Textfensters bringen.



In sekundenschnelle ist dieses Bild aufgebaut

JOYSTICK ve
(ve = Verzögerung. Mit ve = 0 malt man am schnellsten, mit ve = 255 am langsamsten.)

Dieser Befehl erlaubt das Zeichnen von Bildern mit dem Joystick (Port 2). Mit dem Joystick kann der Grafikcursor bewegt werden. Drückt man gleichzeitig den Feuerknopf, wird entsprechend dem eingestellten Modus ein Punkt gesetzt, gelöscht etc. Nun können beliebige Bilder gezeichnet werden, solange, bis mit der F7-Taste mit der Programmabarbeitung fortgefahren wird.

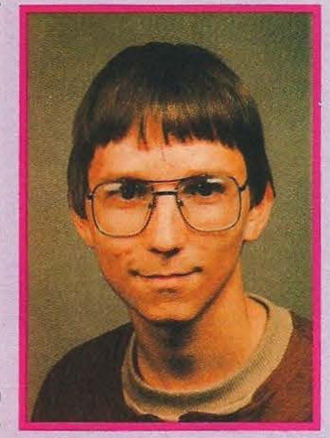
LPEN
Dieser Befehl erlaubt das Zeichnen von Bildern mit dem Lightpen (Port 1). Im Prinzip gilt das gleiche wie beim JOYSTICK-Befehl, nur daß hier statt des Feuerknopfes die CTRL-Taste beziehungsweise der Knopf am Lightpen benutzt wird. Man verläßt den Lightpenmodus mit F7.

Alle Befehle können unabhängig vom Einschaltzustand des Grafikbildschirms angewendet werden. Sie lassen sich auch wie normale Basic-Befehle abkürzen.

Die Funktionen und Anwendungen des Programms und seiner Befehle sind sehr gut in dem Programm »TURTLE DEMO« demonstriert (Listing 2). (Peter Menke/gk)



Der Programmierer der Schildkröte



Geboren wurde ich am 3.1.1968 in Lüneburg. Meine erste Begegnung mit dem Computer fand am dortigen Gymnasium statt. An einem heute schon fast fossilen CBM 4016 lernte ich Basic. Kurz nachdem der Commodore 64 auf den Markt kam, erstand ich ein Gerät für sage und schreibe 1298 Mark (Wucher!!). Wegen seines schwachen Basics lernte ich bald Maschinensprache. Doch das bloße Programmieren von Videospielen befriedigte bald nicht mehr. So stürzte ich mich in die Tiefen des Basic- und Betriebssystem-ROMs, um eine Basic-Befehlserweiterung zu schreiben. Eins der besten und umfangreichsten Ergebnisse: Diese Turtle-Grafik.

(Peter Menke/gk)

TURTLE DEMO BY PETER MENKE

Text und Grafik können gleichzeitig auf dem Bildschirm erscheinen

Das Programm beginnt mit Titel und Autorenanschrift. Das GOSUB in Zeile 260 dient nur zur Suche von Syntax-Fehlern in den DATA-Zeilen. Im Programmteil »Variable« ab Zeilennummer 290 wird der Variablen AN die Startadresse des Maschinenprogramms zugewiesen, der Variable ZI die Endadresse. In NA$ steht der Programmname. Die eigentliche Einleseroutine (ab Zeile 350) funktioniert folgendermaßen: Es werden in einer Schleife die ersten 16 Zahlen aus einer jeden DATA-Zeile gelesen und in den Speicher gePOKEt. Gleichzeitig wird aus den gelesenen Daten eine Prüfsumme gebildet. Diese Summe wird mit der letzten Zahl in der DATA-Zeile verglichen, dies ist die richtige Prüfsumme. Unterscheiden sich die beiden Zahlen, so wurde ein Tippfehler gemacht und die fehlerhafte Zeile wird vom Programm automatisch geLISTet. Außerdem wird geprüft, ob eine Zeile vergessen wurde (Zeile 517-520) und ob die Anzahl der Daten richtig ist (Zeile 530-537). Zusätzlich wird noch getestet, ob die gelesene Zahl auch zwischen 0-255 liegt (Zeile 391). Ist dies nicht der Fall, wird eine Fehlermeldung ausgegeben (Zeile 503-505). Ursache ist wahrscheinlich ein Kommafehler. Wurde kein Fehler gefunden, so fragt das Programm, ob es sich selbst abspeichern soll (davon sollte bei der ersten Benutzung des Programms unbedingt Gebrauch gemacht werden, Zeile 546-580). Danach wird das Maschinenprogramm gestartet (Zeile 590-610). Trotz der Prüfsummen ist ein Fehler in den DATAs nicht völlig ausgeschlossen. Vertauschungen werden zum Beispiel nicht bemerkt. Solche Fehler sind jedoch sehr unwahrscheinlich.

Diese Art der Überprüfung von DATA-Werten sollten Sie sich genau ansehen. Sie erleichtert der Redaktion und vor allem den Lesern die Eingabe und Überprüfung großer Zahlenkolonnen. Falls Sie uns Programme einschicken wollen, dann nehmen Sie sich doch auch bitte etwas Zeit und fügen eine komfortable Prüfroutine in Ihre Programme ein.

Programmierung

a) Speicheraufteilung:
Das Video-RAM steht ab $0400. Der Hires-Speicher wurde hinter $ CC00 gelegt, das Hires-Farb-RAM nach C800. Das Programm selbst beginnt bei $C000 und endet bei $C88B. Programmvariable liegen im Bereich vor $CC00.

b) Die Belegung der Funktionstasten wird durch ein »Anzapfen« des Interrupt ermöglicht. Die Funktion PRINT "(fs)" wird durch Verändern der BSOUT-Routine erreicht.

c) Das Textfenster:
Wie Sie wissen, wird das Bild auf dem Fernseher (Monitor) durch einen Elektronenstrahl erzeugt, der den Bildschirm zeilenweise von oben nach unten abfährt. Der VIC bietet nun die Möglichkeit bei einer bestimmten Zeilenposition einen Interrupt auszulösen. Diese Fähigkeit des VIC wird ausgenutzt. In der dann ausgeführten Interruptroutine wird zwischen Hires- und Textmodus hin und her geschaltet, so daß der Bildschirm in ein Text und ein Grafikfenster eingeteilt wird.

d) Die neuen Befehle:
Es gibt zahlreiche Möglichkeiten, neue Basic-Befehle zu implementieren. Anzapfen der
1. CHRGET-Routine (DOS 5.1)
2. Eingabe-Warteschleife (Toolkits)
3. Interpreterschleife (beschrieben im 64 Intern)

Fortsetzung auf Seite 55

Benutzte Variable:

AN	= Anfangsadresse des Maschinenprogramms
ZI	= Endadresse des Maschinenprogramms
NA$	= Programmname beim Abspeichern
GE	= Geräteadresse beim Abspeichern
X	= eingelesener DATA-Wert
S	= Speicherstelle, in die X gePOKEt wird
Z	= aktuelle DATA-Zeilennummer
PR	= errechnete Prüfsumme
I	= Variable der FOR/NEXT-Einleseschleife



Der Schachmeister und sein Programm

Ich bin 16 Jahre alt und gehe in die 11. Klasse eines Ulmer Gymnasiums. Seit Weihnachten '83 besitze ich einen Commodore 64; sein Vorgänger war ein ZX81.
Das Programm:
Der Computer spielt berühmte oder weniger berühmte Schachpartien vor, die man leicht selbst einprogrammieren kann, so daß man sich mit der Zeit eine richtige Sammlung aufbaut. Besonderer Wert wurde auf eine gute Multicolor-Grafik gelegt.
Entstehungsgeschichte:
Das Programm entstand aus einer spontanen Idee heraus. In Büchereien findet man unzählige Bücher mit aufgeschriebenen Schachpartien, die man auf einem Brett nachspielen kann, was aber nicht unbedingt jedermanns Sache ist, und außerdem ist die Gefahr groß, daß man sich verschaut und einen Fehler macht. Da müßte es doch viel einfacher sein, wenn der Computer dies übernehmen würde. In zirka einwöchiger Arbeit entstand so das vorliegende Programm.
Wesentlich Merkmale und Vorzüge des Programms:
— Multicolorgrafik, die sich zum Beispiel hinter »Grandmaster« nicht zu verstecken braucht.
— alle wichtigen Schachregeln sind berücksichtigt (zum Beispiel Rochaden, Dametausch, Matt, Remis ...)
— jeder Benutzer kann seine eigenen Partien eingeben und mit dem Programm speichern (maximal 20)
— das Bild kann jederzeit »eingefroren« werden
— der Programmablauf kann beschleunigt werden.

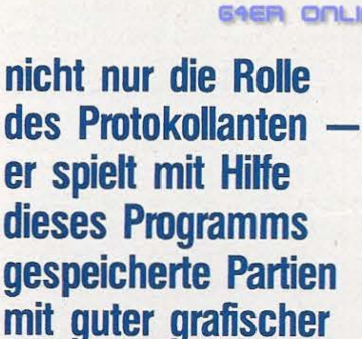
(Thomas Behrend)





SCHACH

Viele Schachspieler zeichnen ihre Partien auf und überprüfen sie hinterher. Oder sie spielen berühmte Partien nach. Der C 64 übernimmt nicht nur die Rolle des Protokollanten — er spielt mit Hilfe dieses Programms gespeicherte Partien mit guter grafischer Darstellung nach.





Nach dem Start des Programms muß man einen Augenblick warten, bis der Computer die neuen Multicolor-Zeichen definiert hat und diverse Felder und Listen aufgestellt hat (zirka 20 Sekunden). Darauf kann man eine der gespeicherten Partien auswählen und das Schachbrett baut sich auf.

Am rechten Rand werden die Namen der Spieler, das Jahr, die Zugnummer und der Zug sowie verschiedene Meldungen wie »Schach«, »Matt«, »Rochade« ... ausgegeben. Das Schachbrett nimmt zirka ¾ des Bildschirms ein.

Durch Drücken der F7-Taste nach einem Zug wird das Bild bis zu einem erneuten Tastendruck »eingefroren«. Durch Drücken einer anderen Taste nach einem Zug wird der Ablauf beschleunigt. Durch anhaltendes Drücken der Space-Taste kann man so eine Art »Blitzschach« erreichen. Nach Spielende führt ein Tastendruck wieder zum Auswahlmenü.

Eingeben eigener Schachpartien:

1. Zuerst werden die Namen der Spieler und das Jahr eingegeben:
10000 DATA SPIELER 1, SPIELER 2, 1984.
2. Die Eingabe der Züge:
Beispiel für einen normalen Zug: 10010 DATA E2E4, E7E5, G1F3 ...

Wenn eine Figur eine gegnerische schlägt, braucht man das nicht anzugeben; der Computer erkennt das von selbst.

3. Sonderregeln:

■ Aufgabe eines Spielers:
10020 DATA ...,SA
(Schwarz gibt auf)
10020 DATA ..., WA
(Weiß gibt auf)

■ Rochaden:
10020 DATA ..., KR
(kurze Rochade)
10020 DATA ..., LR
(lange Rochade)
Es braucht nicht angegeben zu werden, welcher der beiden Spieler rochiert.

■ Schach, Matt, Remis:
Um dem Computer einen dieser drei Zustände anzuzeigen, muß ein S (Schach), ein M (Matt) oder ein R (Remis) dem jeweiligen Zug angehängt werden:
10020 DATA ..., D6E6S
(Schach wird geboten)
10030 DATA ..., F6E6M
(Schachmatt)
10020 DATA ..., E5E6R
(Remis)

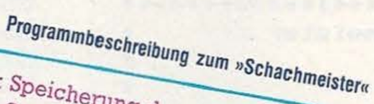
■ Dametausch:
Wenn ein Bauer in eine Dame umgetauscht wird, so ist dem jeweiligen Zug ein D anzufügen:
10020 DATA ..., A7A8D
(weißer Bauer wird in Dame getauscht)

Wichtig:
Nach Ende jeder Partie muß unbedingt ein E angefügt werden, damit der Computer eine Liste der Partien aufstellt. Nach der letzten Partie ist statt dem E ein X anzufügen:
19999 DATA ...,F6E6M,E
19999 DATA ...,F6E6M,X

Die Programmlänge ohne gespeicherte Partien beträgt zirka 7,5 KByte pro gespeicherter Partie werden zusätzlich zirka 0,5 KByte benötigt.

Da der Bildschirmspeicher verschoben wird, muß nach einem RUN STOP-RESTORE blind RUN eingetippt werden, um das Programm wieder auflisten zu lassen.

(Thomas Behrend/rg)



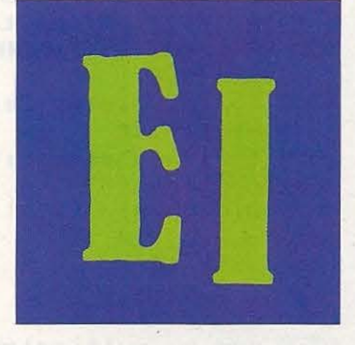
Programmbeschreibung zum »Schachmeister«

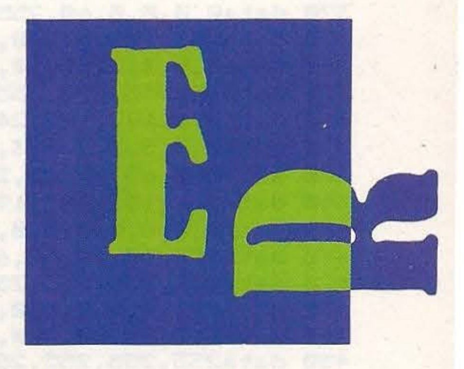
A$-Feld	: Speicherung der sechs Schachfiguren
A(x,y)-Feld	: Spielfeld (Figuren)
B(x,y)-Feld	: Spielbrett
P-Feld	: Zähler zum Überspringen der DATA-Zeilen bis zur gewählten Partie
P$-Feld	: Namen der Spieler (weiß)
P1$-Feld	: Namen der Spieler (schwarz)
P	: Anzahl der gespeicherten Partien
DA	: Zähler zum Überspringen der ersten DATA-Zeilen
AW	: Flag für verschiedene Spielzustände (Matt, Remis...)

Wichtige Programmzeilen und Programmteile:

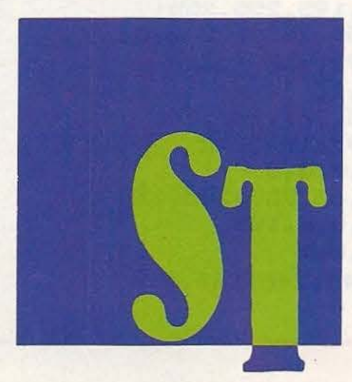
150- 190	: MC-Programm zur Beschleunigung der Zeichendefinition
200- 260	: Zeichendefinition
300- 860	: Zeichen-DATA (Multicolor)
900-1050	: Felder aufstellen und Variablen definieren
1100,1110	: Multicolor-Modus einschalten
1170-1250	: Partienanfänge einlesen und Tabelle aufstellen
1290-1350	: Auswahlmenü
1390-1660	: Bildschirmaufbau
1730-1910	: Hauptprogramm (normaler Zug)
1950-2410	: Besondere Regeln (Rochade, Dametausch, Matt, Remis...)
2470-2490	: Einzelnes Feld löschen
2530-2580	: Figur drucken
2620	: Koordinaten errechnen
2660-2720	: Zeitschlaufe mit Unterbrechung und Verkürzung
2760-2800	: Grundstellung der Figuren
2840-2850	: Bildschirmpositionierung
2890-2910	: Spielende
10000-	: DATAs für die Partien

Alle REM-Zeilen dienen nur zur Anschaulichkeit und können entfernt werden.









Listing »Schachmeister«

```
10 rem **********************************
20 rem *     >> schachmeister <<       *
30 rem *                               *
40 rem * thomas behrend                *
50 rem * in der wanne 16    7900 ulm   *
60 rem *       (8.1984)                *
70 rem **********************************
80 rem
90 rem -- alle rem-zeilen koennen    --
100 rem -- weggelassen werden        --
110 rem
120 rem -- neue zeichen --
130 rem
140 poke53281,6:poke53280,1:print"■"chr$
(5)
150 data120,169,51,133,1,169,0,133,95,13
3,90,133,88,169,208,133,96,169,240,133
160 data89,169,224,133,91,32,191,163,169
,55,133,1,88,96
170 rem -- mc-programm --
180 j=4097:fori=832to865:reada:j=j-a:pok
ei,a:next
190 ifj<>0thenprint"data-fehler in zeile
 15-20":stop
200 sys832:poke850,160:sys832:poke56576,
peek(56576)and252:poke53272,8
210 poke648,192:print"■"tab(5)"> schachm
eister <"
220 print"■■"tab(5)"bitte einen moment w
arten..."
230 ad=57344:fori=64to91:b=ad+i*8:forj=0
to7:readc:pokeb+j,c:next:next
240 fori=192to219:b=ad+i*8:forj=0to7:rea
dc:pokeb+j,c:next:next
250 i=175:b=ad+i*8:forj=0to7:readc:pokeb
+j,c:next
260 poke56,130
270 rem
280 rem -- data fuer schachfiguren --
290 rem
300 data0,0,60,60,255,255,60,60
310 data0,0,0,0,0,0,0,0
320 data0,0,0,0,60,255,255,255
330 data255,255,60,60,60,60,60,60
340 data0,0,3,3,15,10,0,0
350 data60,255,255,255,255,170,0,0
360 data0,0,192,192,240,160,0,0
370 data0,0,3,3,3,3,3,3
380 data0,0,60,60,60,255,255,255
390 data0,0,192,192,192,192,192,192
400 data3,0,0,0,0,0,0,0
410 data255,60,60,60,60,60,60,60
420 data255,255,255,255,255,255,255,255
430 data192,0,0,0,0,0,0,0
440 data0,0,60,60,240,195,255,255
450 data255,255,255,255,255,170,0,0
460 data0,0,0,0,192,192,0,0
470 data0,0,0,0,0,3,3,15
480 data0,48,48,63,255,207,207,255
490 data0,0,0,0,192,192,192,192
500 data63,63,63,60,0,0,0,3
510 data255,207,15,63,63,255,255,255
520 data192,192,0,0,0,0,0,192
530 data3,3,3,3,15,10,0,0
540 data192,240,240,240,240,160,0,0
550 data0,0,12,15,3,3,3,3
560 data0,0,48,240,192,192,192,192
570 data0,0,0,0,3,3,0,0
580 data85,85,125,125,255,255,125,125
590 data85,85,85,85,85,85,85,85
600 data85,85,85,85,125,255,255,255
610 data255,255,125,125,125,125,125,125
620 data85,85,87,87,95,90,85,85
630 data125,255,255,255,255,170,85,85
640 data85,85,213,213,245,165,85,85
650 data85,85,87,87,87,87,87,87
660 data85,85,125,125,125,255,255,255
670 data85,85,213,213,213,213,213,213
680 data87,85,85,85,85,85,85,85
690 data255,125,125,125,125,125,125,125
700 data255,255,255,255,255,255,255,255
710 data213,85,85,85,85,85,85,85
720 data85,85,125,125,245,215,255,255
730 data255,255,255,255,255,170,85,85
740 data85,85,85,85,213,213,85,85
750 data85,85,85,85,85,87,87,95
760 data85,117,117,127,255,223,223,255
770 data85,85,85,85,213,213,213,213
780 data127,127,127,125,85,85,85,87
790 data255,223,95,127,127,255,255,255
800 data213,213,85,85,85,85,85,213
810 data87,87,87,87,95,90,85,85
820 data213,245,245,245,245,165,85,85
830 data85,85,93,95,87,87,87,87
840 data85,85,117,245,213,213,213,213
850 data85,85,85,85,87,87,85,85
860 data126,189,215,231,231,215,189,126
870 rem
880 rem -- variablen und felder --
890 rem
900 b$(1)="abaacadef":rem bauer
910 b$(2)="ghijlmdof":rem turm
920 b$(3)="anaakadef":rem laeufer
930 b$(4)="qrstuvwox":rem pferd
940 b$(5)="yhzacadef":rem dame
950 b$(6)="[@pjcmdef":rem koenig
960 fori=1to6:forj=0to2:fork=1to3
970 a$=mid$(b$(i),j*3+k,1):a$(i)=a$(i)+c
hr$(asc(a$)+128):next
980 a$(i)=a$(i)+"■■■■■":next:next
990 dimb(8,8)
1000 fori=7to1step-2:forj=1to7step2:b(i,
j)=1:next:next
1010 fori=8to2step-2:forj=2to8step2:b(i,
j)=1:next:next
1020 dima(8,8)
1030 gosub2760
1040 z1$="■■ ■                    "
1050 :
1060 rem
1070 rem -- multicolor-modus --
1080 rem -- einschalten      --
1090 rem
1100 poke53281,6:poke53282,7:poke53283,5
1110 poke53270,peek(53270)or16
1120 rem
1130 rem -- liste der partien --
1140 rem -- aufstellen        --
1150 rem
1160 data-1
```

```
1170 restore:da=0
1180 da=da+1:reada:ifa=-1then1200
1190 goto1180
1200 dimp(20),p$(20),p1$(20),ja$(20):p=0
:d=0
1210 p=p+1:readp$(p),p1$(p),ja$(p):d=d+3
1220 reada$:d=d+1:ifa$="e"ora$="x"then12
40
1230 goto1220
1240 ifa$="x"then1290
1250 p(p)=d:goto1210
1260 rem
1270 rem -- auswahlmenue --
1280 rem
1290 restore:fori=1toda:reada:next
1300 printchr$(5)"■■    - auswahl -"
1310 print:print
1320 fori=1top:printi;tab(5)p$(i)tab(16)
p1$(i),ja$(i):next
1330 poke198,0:input"■■";w:ifw<1orw>pthe
n1330
1340 print"■":ifw=1then1390
1350 fori=1top(w-1):reada$:next
1360 rem
1370 rem -- spielbrett --
1380 rem
1390 fori=1to24:printchr$(159)tab(0)"■ "
tab(25)z1$:next
1400 printchr$(5)"■";
1410 fory=8to1step-1:x=1:gosub2840:print
"■■■"y:x=8:gosub2840
1420 print"■■■"y:next
1430 fory=8to1step-1:forx=1to8
1440 gosub2470
1450 next:next
1460 print"■"
1470 rem
1480 rem -- anfangsstellung --
1490 rem
1500 fory=7to8:forx=1to8
1510 gosub2530:next:next
1520 fory=1to2:forx=1to8:gosub2530:next:
next
1530 printchr$(159)"■":printtab(27)"■ sp
ieler 1 "
1540 printtab(27)"■         ■"
1550 printtab(27)"■■         "
1560 poke214,18:print:printtab(27)chr$(3
0)"■ spieler 2 "
1570 printtab(27)"■         ■"
1580 printtab(27)"■■         "
1590 poke214,5:print:printtab(27)chr$(5)
"■zug:■  1"
1600 poke214,7:print:printtab(28)chr$(15
8)"■        ■"
1610 poke214,9:print:printtab(27)chr$(5)
"■         "
1620 printtab(27)"■         "
1630 printtab(27)"■■         "
1640 reada$:a$=left$(a$,9):poke214,20:pr
int:printtab(28)"■"a$
1650 reada$:a$=left$(a$,9):poke214,2:pri
nt:printtab(28)"■"a$
1660 reada$:poke214,15:print:printtab(28
)a$
1670 zu=1:z1=-1:aw=0
1680 rem
1690 rem **********************
1700 rem *  -- hauptprogramm --  *
1710 rem **********************
1720 rem
1730 print"■":z1=z1+1:ifz1=2thenz1=0:zu=
zu+1:poke214,5:print:printtab(32)chr$(5)
zu
1740 reada$:iflen(a$)<>4then1950
1750 rem
1760 rem -- normaler zug --
1770 rem
1780 b$=left$(a$,2):gosub2620:x=x2:y=y2
1790 b2$=b$
1800 b$=right$(a$,2):gosub2620
1810 o$="-":ifa(x2,y2)<>0theno$="/"
1820 a$=b2$+o$+b$
1830 poke214,7:print:printtab(29)chr$(5)
"■"a$
1840 fori=1to4:gosub2530:gosub2470:next
1850 a1=a(x,y):a(x,y)=0:a(x2,y2)=a1
1860 x=x2:y=y2:fori=1to4:gosub2470:gosub
2530:next
1870 ifaw=3thenaw=0:return
1880 ifaw=2then2890:rem-matt oder remis-
1890 gosub2660
1900 ifaw=1thenaw=0:poke214,11:print:pri
nttab(27)chr$(5)"■          "
1910 goto1730
1920 rem
1930 rem -- besondere situationen --
1940 rem
1950 ifa$="e"then2890
1960 ifa$="sa"then2100
1970 ifa$="wa"then2120
1980 ifa$="kr"then2190
1990 ifa$="lr"then2250
2000 printchr$(5)
2010 b$=right$(a$,1):a$=left$(a$,4)
2020 ifb$="s"thenaw=1:poke214,11:print:p
rinttab(27)"■schach !":goto1780
2030 ifb$="m"thenaw=2:poke214,11:print:p
rinttab(27)"■ matt !":goto1780
2040 ifb$="r"thenaw=2:poke214,11:print:p
rinttab(27)"■ remis !":goto1780
2050 ifb$="d"then2340
2060 print"■datafehler !!!":stop
2070 rem
2080 rem -- aufgeben --
2090 rem
2100 a$="■schwarz"
2110 goto2130
2120 a$="■weiss"
2130 poke214,9:print:printtab(29)a$
2140 printtab(27)"■gibt auf !"
2150 goto2890
2160 rem
2170 rem -- kurze rochade --
2180 rem
2190 y=1:t=0:ifz1=1theny=8:t=6
2200 a(5,y)=0:a(8,y)=0:a(6,y)=8-t:a(7,y)
=12-t:x=5:gosub2470:x=8:gosub2470
2210 x=6:gosub2530:x=7:gosub2530:goto228
0
```

Listing »Schachmeister« (Fortsetzung)

```
2220 rem
2230 rem -- lange rochade --
2240 rem
2250 y=1:t=0:ifz1=1theny=8:t=6
2260 a(1,y)=0:a(5,y)=0:a(3,y)=12-t:a(4,y
)=8-t
2270 x=1:gosub2470:x=5:gosub2470:x=3:gos
ub2530:x=4:gosub2530
2280 poke214,11:print:printtab(29)chr$(5
)"█rochade"
2290 gosub2660
2300 poke214,11:print:printtab(29)"█
   ":goto1730
2310 rem
2320 rem -- dametausch --
2330 rem
2340 aw=3:gosub1780
2350 ify2=8thena(x2,8)=11:goto2370
2360 a(x2,1)=5
2370 x=x2:y=y2:gosub2530
2380 poke214,10:print:printtab(29)chr$(5
)"█dame-"
2390 printtab(29)"█tausch"
2400 gosub2660:poke214,10:print:printtab
(29)"█      "
2410 printtab(29)"█      ":goto1730
2420 rem ****** unterprogramme ******
2430 rem
2440 rem -- einzelne felder --
2450 rem -- drucken         --
2460 rem
2470 ifb(x,y)=1then2490
2480 gosub2840:poke646,15:print"█AAA███
AAA███AAA":return
2490 gosub2840:print"    ████   ████   ":
return
2500 rem
2510 rem -- figur drucken --
2520 rem
2530 f=a(x,y):iff=0thenreturn
2540 gosub2840
2550 poke646,9:iff>6andb(x,y)=0then:prin
t"█"a$(f-6)"█":return
2560 iff>6andb(x,y)=1then:print"█"a$(f-6
)"█":return
2570 poke646,8:ifb(x,y)=1thenprint"█"a$(
f)"█":return
2580 print"█"a$(f)"█":return
2590 rem
2600 rem -- umrechnen --
2610 rem
2620 b1$=left$(b$,1):x2=asc(b1$)-64:y2=v
al(right$(b$,1)):return
2630 rem
2640 rem -- zeitschlaufe --
2650 rem
2660 poke198,0:fort=1to250:ifpeek(197)<>
64then2680
2670 next:return
2680 ifpeek(197)=3then2700
2690 return
2700 ifpeek(197)<>64then2700
2710 ifpeek(197)=64then2710
2720 return
2730 rem
2740 rem -- grundstellung --
2750 rem
2760 a(1,1)=8:a(2,1)=10:a(3,1)=9:a(4,1)=
11:a(5,1)=12:a(6,1)=9
2770 a(7,1)=10:a(8,1)=8
2780 fori=1to8:a(i,8)=a(i,1)-6:next
2790 forx=1to8:a(x,2)=7:a(x,7)=1:next
2800 return
2810 rem
2820 rem -- bildschirmposition --
2830 rem
2840 ify=8thenprint"█"tab((x-1)*3+1);:re
turn
2850 x1=(x-1)*3+1:y1=(8-y)*3-1:poke214,y
1:print:printtab(x1);:return
2860 rem
2870 rem -- spielende ---
2880 rem
2890 poke198,0:fori=1to10000:ifpeek(197)
=64thennext
2900 fori=1to8:forj=1to8:a(i,j)=0:next:n
ext
2910 gosub2760:goto1290
10000 rem
10010 rem -----  data fuer die  ------
10020 rem -----  partien        ------
10030 rem
10040 rem ** demo-partie 1 **
10050 datasiff,kashdan,1948
10060 datad2d4,g8f6,c2c4,e7e6,b1c3,f8b4,
d1c2,d7d5,a2a3,b4e7,c4d5,e6d5,c1f4,c7c6
10070 datah2h3,kr,e2e3,f8e8,f1d3,b8d7,g1
f3,d7f8,f3e5,e7d6,kr,f8e6,f4h2,g7g6
10080 datae5f3,e6g7,h2d6,d8d6,f1c1,c8f5,
f3d2,e8e7,b2b4,a8e8,a1b1,f5h3,g2h3,e7e3
10090 datad3f1,g7f5,f2e3,d6g3s,f1g2,g3e3
s,g1h1,f5g3s,h1h2,e3f4,d2f3,e8e2,c3e2
10100 datag3e2s,h2h1,f6h5,c2d2,h5g3s,h1h
2,g3f1s,h2h1,f4h2s,f3h2,f1g3m
10110 datae
10120 rem ** demo-partie 2 **
10130 datageller,euwe,1953
10140 datad2d4,g8f6,c2c4,e7e6,b1c3,f8b4,
e2e3,c7c5,a2a3,b4c3,b2c3,b7b6,f1d3,c8b7
10150 dataf2f3,b8c6,g1e2,kr,kr,c6a5,e3e4
,f6e8,e2g3,c5d4,c3d4,a8c8,f3f4,a5c4
10160 dataf4f5,f7f6,f1f4,b6b5,f4h4,d8b6,
e4e5,c4e5,f5e6,e5d3,d1d3,b6e6,d3h7s
10170 datag8f7,c1h6,f8h8,h7h8,c8c2,a1c1,
c2g2s,g1f1,e6b3,f1e1,b3f3,wa
10180 datae
10190 rem ** demo-partie 3 **
10200 datarossetto,stahlberg,1947
10210 datae2e4,e7e6,d2d4,d7d5,b1c3,g8f6,
c1g5,f8e7,e4e5,f6d7,h2h4,c7c5,c3b5,f7f6
10220 dataf1d3,a7a6,d1h5s,e8f8,h1h3,a6b5
,g5h6,d8a5s,h6d2,a5c7,h3g3,c5d4,g1f3
10230 datad7e5,g3g7,h7h6,d3h7,f8g7,h5h6s
,g7f7,h6h5s,f7g7r
10240 datax

ready.
```

Listing »Schachmeister« (Schluß)



Die schnelle Schildkröte Turtle-Grafik

Fortsetzung der Beschreibung von Seite 49, Einzelheiten zur Programmierung und das Programmlisting

Die drei auf Seite 49 genannten Möglichkeiten haben jedoch alle große Nachteile: Sie sind alle entweder zu langsam (1. und 3.), die Befehle lassen sich nur im Direktmodus anwenden (2.) oder sie können nicht abgekürzt werden. Deshalb wurden die Befehle mit sehr komplexen Routinen voll als Tokens integriert. Aus diesem Grund wurde die Interpreterschleife, die Routine zur Umwandlung in Interpretercode und die Routine zur Rückumwandlung in Klartext verändert. Die Vorteile dieser Arbeitsweise sind, daß sich die neuen Befehle wie normale Basic-Befehle anwenden lassen und daß nur minimale Geschwindigkeitsverluste bei der Ausführung von Basic-Programmen in Kauf genommen werden müssen. Ähnliche Methoden verwenden übrigens auch professionelle Basic-Erweiterungen, wie zum Beispiel Simons Basic.

Ein solches »Anzapfen« von Betriebssystem- und Basic-Routinen wird beim C 64 durch zahlreiche Sprungvektoren ermöglicht, die verändert werden können. Dies ist leider nicht bei allen Computern so gut möglich. Die hier benutzten Vektoren sind:

Name:	Adresse in dezimal:
Interrupt-Vektor	: 788/789
NMI-Vektor	: 792/793
BRK-Vektor	: 790/791
Bsout-Vektor	: 806/807
Vektor für Umwandlung in Interpretercode	: 772/773
Vektor für Umwandlung in Klartext	: 774/775
Vektor für Basic-Befehl ausführen (Interpreterschleife)	: 776/777

Das Wissen über die Programmierung des Programms ist für dessen Anwendung nicht nötig. Doch nun viel Spaß mit Turtle Grafik. (Peter Menke/gk)

```
20 PRINT"▮▮▮     "▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚
30 PRINT"     ▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚
40 PRINT"     ▚▚
 ▚▚
50 PRINT"     ▚▚
 ▚▚
60 PRINT"     ▚▚     TURTLE GRAPHICS
 ▚▚
70 PRINT"     ▚▚     _______________
 ▚▚
100 PRINT"     ▚▚
  ▚▚
110 PRINT"     ▚▚   ENTWICKELT 8.1984 VON
  ▚▚
120 PRINT"     ▚▚
  ▚▚
130 PRINT"     ▚▚
  ▚▚
140 PRINT"     ▚▚    P E T E R   M E N K E
  ▚▚
150 PRINT"     ▚▚
  ▚▚
160 PRINT"     ▚▚         GRIFTWEG 4
  ▚▚
170 PRINT"     ▚▚     2212 BRUNSBUETTEL
  ▚▚
190 PRINT"     ▚▚
  ▚▚
200 PRINT"     ▚▚
  ▚▚
210 PRINT"     ▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚
220 PRINT"     ▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚
230 PRINT"▮▮          BITTE TASTE DRUECKE
N"
240 GET A$:IF A$=""THEN240
250 PRINT"▮"
260 GOSUB1000
270 :
280 :
290 REM **VARIABLE**
300 AN=49152
301 ZI=51360
310 NA$="TURTLE GR."
320 :
330 :
340 :
350 REM **** EINLESEROUTINE ****
360 S=AN:Z=1000
370 FORI=1TO16:READ X
390 IFX=-100THEN530
391 IFX<0ORX>255THEN503
400 GOSUB515:POKES,X
410 S=S+1:PR=PR+X:NEXT
440 :
460 READ X:IF PR=X THEN PR=0:Z=Z+10:GOTO
 370
470 PRINT"▮▮PRUEFSUMMENFEHLER IN ZEILE"Z
475 PRINT"▮PRUEFSUMME DER ZEILE"Z":"PR:P
RINT
480 PRINT"RICHTIGE PRUEFSUMME:"X:PRINT:P
RINT
490 PRINT"▮LIST"Z-10"-"Z"▮▮";:POKE631,13
:POKE198,1
500 END
501 :
502 :
503 PRINT"DATENFEHLER IN ZEILE"Z
504 PRINT"ES WURDE VERSUCHT"X"ZU POKEN."
```

Listing Turtle-Grafik

```
505 GOTO490
510 :
511 :
512 :
515 PRINT"▒▒▒  ZEILE:"Z" SPEICHERSTELLE:
"S
516 PRINT"▒  EINGELESENER WERT:      ■■■■■"
X
517 IF Z>=PEEK(63)+256*PEEK(64) THEN RET
URN
518 PRINT"▒▒▒DATA-ZEILE FEHLT!!"
519 PRINT"ODER ZEILENNUMMERN DER DATA-ZE
ILEN NICHT"
520 PRINT"IN ZEHNER ABSTAENDEN":END
521 :
522 :
523 :
530 IF S=ZI THEN 539
531 PRINT"▒▒▒DATENANZAHL IST FALSCH":PRI
NT:PRINT"DAS WAERE RICHTIG :";ZI-AN
532 PRINT:PRINT"IHRE DATENANZAHL :";S-AN
537 END
539 PRINT"▒▒KEINEN FEHLER GEFUNDEN !▒"
540 PRINT"ABSPEICHERN (J/N) ?"
550 GET A$:IF A$="N" THEN 590
560 IF A$<>"J" THEN 550
570 INPUT"GERAETEADRESSE (1=DATASETTE/8=
DISK)";GE
580 SAVE NA$+"(C)PM",GE
590 PRINT"▒ZUM START EINE BELIEBIGE
          TASTE DRUECKEN
600 GET A$:IF A$="" THEN 600
610 SYSAN
630 :
640 :
650 :

1000 DATA 120,032,063,193,088,032,163,19
3,032,223,193,169,028,133,254,169,2085
1010 DATA 192,133,255,032,097,192,032,06
8,166,076,116,164,147,017,017,017,1721
1020 DATA 029,029,084,085,082,084,076,06
9,032,071,082,065,080,072,073,067,1080
1030 DATA 083,190,190,029,029,066,089,03
2,080,069,084,069,082,032,077,069,1270
1040 DATA 078,075,069,190,190,029,029,05
1,056,057,049,049,032,066,065,083,1168
1050 DATA 073,067,032,066,089,084,069,08
3,032,070,082,069,069,017,017,190,1109
1060 DATA 191,160,000,177,254,201,191,24
0,012,201,190,208,002,169,013,032,2241
1070 DATA 210,255,200,208,238,096,173,02
5,208,141,025,208,016,029,173,018,2223
1080 DATA 208,201,250,176,011,032,204,19
3,169,250,141,018,208,076,188,254,2579
1090 DATA 032,185,193,169,217,141,018,20
8,076,188,254,173,013,220,088,165,2340
1100 DATA 204,208,033,169,032,044,017,20
8,240,026,206,252,203,208,021,169,2240
1110 DATA 007,141,252,203,173,241,203,07
2,169,002,141,241,203,032,118,195,2393
1120 DATA 104,141,241,203,165,212,208,03
0,165,216,208,026,165,203,205,255,2747
1130 DATA 203,240,019,141,255,203,201,06
4,240,012,201,004,240,011,201,005,2240
1140 DATA 240,029,201,006,240,048,076,04
9,234,173,017,208,041,032,240,009,1843
1150 DATA 032,204,193,032,017,194,076,23
0,192,032,185,193,076,230,192,173,2251
1160 DATA 017,208,041,032,240,246,173,02
6,208,041,001,240,006,032,017,194,1722
1170 DATA 076,230,192,032,254,193,162,02
1,160,000,024,032,240,255,076,230,2177
1180 DATA 192,072,138,072,152,072,169,12
7,141,013,221,172,013,221,048,012,1835
1190 DATA 032,163,253,032,024,229,032,06
3,193,108,002,160,076,114,254,032,1767
1200 DATA 204,193,169,004,141,136,002,16
9,118,141,020,003,169,192,141,021,1823
1210 DATA 003,169,033,141,024,003,169,19
3,141,025,003,169,048,141,022,003,1287
1220 DATA 169,193,141,023,003,169,011,14
1,254,203,169,001,141,253,203,141,2215
1230 DATA 252,203,169,000,141,026,208,16
9,134,141,004,003,169,196,141,005,1961
1240 DATA 003,169,043,141,008,003,169,19
7,141,009,003,169,084,141,006,003,1289
1250 DATA 169,197,141,007,003,169,103,14
1,038,003,169,199,141,039,003,032,1554
1260 DATA 249,195,096,162,032,160,000,16
9,224,133,255,132,254,152,145,254,2612
1270 DATA 136,208,251,230,255,202,208,24
6,096,169,056,141,024,208,169,148,2747
1280 DATA 141,000,221,173,017,208,009,03
2,141,017,208,096,169,021,141,024,1618
1290 DATA 208,169,151,141,000,221,173,01
7,208,041,223,141,017,208,096,169,2183
1300 DATA 204,133,255,160,000,132,254,16
2,004,173,253,203,010,010,010,010,1973
1310 DATA 013,254,203,145,254,136,208,25
1,230,255,202,208,246,096,169,217,3087
1320 DATA 141,018,208,173,017,208,041,12
7,141,017,208,169,001,141,026,208,1844
1330 DATA 096,169,000,141,026,208,096,16
6,122,202,232,142,242,203,162,255,2462
1340 DATA 142,251,203,142,250,203,238,25
0,203,174,250,203,189,127,194,240,3259
1350 DATA 033,016,243,238,251,203,172,24
2,203,136,200,185,000,002,141,249,2514
1360 DATA 203,232,189,127,194,056,237,24
9,203,240,239,201,128,240,019,076,2833
1370 DATA 038,194,174,242,203,189,000,00
2,240,007,201,058,240,188,232,208,2416
1380 DATA 244,096,174,242,203,173,251,20
3,024,105,208,157,000,002,232,200,2514
1390 DATA 185,000,002,157,000,002,240,21
8,232,200,208,244,076,082,194,200,2240
1400 DATA 072,073,082,069,211,077,079,08
6,197,067,076,069,065,210,087,073,1593
1410 DATA 078,068,079,215,080,076,079,21
2,077,079,068,197,068,069,199,082,1726
1420 DATA 069,086,069,082,211,076,084,08
5,082,206,082,084,085,082,206,067,1656
1430 DATA 079,076,079,210,071,076,079,06
5,196,071,083,065,086,197,074,079,1586
1440 DATA 089,083,084,073,067,203,076,08
0,069,206,072,069,076,208,000,239,1694
```

Listing. Turtle-Grafik (Fortsetzung)

```
1450 DATA 194,205,197,095,195,084,199,07
3,196,118,196,147,197,163,197,128,2584
1460 DATA 198,144,198,165,198,036,199,22
3,198,161,199,059,200,053,199,032,2462
1470 DATA 158,183,224,000,240,054,032,12
1,000,240,040,032,253,174,032,158,1941
1480 DATA 183,224,016,144,003,076,106,19
5,142,254,203,032,223,193,032,121,2147
1490 DATA 000,240,016,032,253,174,032,15
8,183,224,016,144,003,076,106,195,1852
1500 DATA 142,032,208,032,185,193,032,24
9,195,076,174,167,032,121,000,240,2078
1510 DATA 037,032,253,174,032,158,183,22
4,016,144,003,076,106,195,142,033,1808
1520 DATA 208,032,121,000,240,016,032,25
3,174,032,158,183,224,016,144,003,1836
1530 DATA 076,106,195,142,032,208,032,01
7,194,032,204,193,076,174,167,208,2056
1540 DATA 012,032,163,193,032,223,193,07
6,174,167,162,014,044,162,011,032,1690
1550 DATA 204,193,076,055,164,096,173,24
1,203,201,003,240,248,174,246,203,2720
1560 DATA 138,074,074,041,254,168,185,01
5,196,141,244,203,185,016,196,141,2271
1570 DATA 245,203,138,041,007,024,109,24
4,203,141,244,203,173,247,203,041,2466
1580 DATA 248,141,243,203,173,244,203,13
3,252,024,169,224,109,245,203,133,2947
1590 DATA 253,024,165,252,109,243,203,13
3,252,165,253,109,248,203,133,253,2998
1600 DATA 173,247,203,041,007,073,007,17
0,189,065,196,160,000,120,162,052,1865
1610 DATA 134,001,174,241,203,240,008,22
4,001,240,014,224,002,240,019,017,1982
1620 DATA 252,145,252,169,055,133,001,08
8,096,073,255,049,252,145,252,076,2293
1630 DATA 227,195,081,252,145,252,076,22
7,195,169,000,141,248,203,141,241,2793
1640 DATA 203,141,240,203,169,160,141,24
7,203,169,100,141,246,203,096,000,2662
1650 DATA 000,064,001,128,002,192,003,00
0,005,064,006,128,007,192,008,000,800
1660 DATA 010,064,011,128,012,192,013,00
0,015,064,016,128,017,192,018,000,880
1670 DATA 020,064,021,128,022,192,023,00
0,025,064,026,128,027,192,028,000,960
1680 DATA 030,001,002,004,008,016,032,06
4,128,032,235,183,224,200,176,035,1370
1690 DATA 165,021,201,001,144,008,208,02
7,165,020,201,064,176,021,120,142,1684
1700 DATA 246,203,165,020,141,247,203,16
5,021,141,248,203,088,032,118,195,2436
1710 DATA 076,174,167,076,106,195,032,15
8,183,224,004,144,003,076,106,195,1919
1720 DATA 142,241,203,076,174,167,032,02
3,194,166,122,160,004,132,015,189,2040
1730 DATA 000,002,141,249,203,041,240,20
1,208,208,006,173,249,203,076,232,2432
1740 DATA 196,173,249,203,016,007,201,25
5,240,062,232,208,226,201,032,240,2741
1750 DATA 055,133,008,201,034,240,086,03
6,015,112,045,201,063,208,004,169,1610
1760 DATA 153,208,037,201,048,144,004,20
1,060,144,029,132,113,160,000,132,1766
1770 DATA 011,136,134,122,202,200,232,18
9,000,002,056,249,158,160,240,245,2336
1780 DATA 201,128,208,048,005,011,164,11
3,232,200,153,251,001,185,251,001,2152
1790 DATA 240,054,056,233,058,240,004,20
1,073,208,002,133,015,056,233,085,1891
1800 DATA 208,141,133,008,189,000,002,24
0,223,197,008,240,219,200,153,251,2412
1810 DATA 001,232,208,240,166,122,230,01
1,200,185,157,160,016,250,185,158,2521
1820 DATA 160,208,180,189,000,002,016,19
0,076,009,166,032,115,000,041,240,1624
1830 DATA 201,208,240,009,032,121,000,03
2,237,167,076,174,167,032,121,000,1817
1840 DATA 041,015,010,168,185,208,194,13
3,167,185,207,194,133,166,032,115,2153
1850 DATA 000.108,166,000,016,052,201,25
5,240,048,036,015,048,044,141,249,1619
1860 DATA 203,041,240,201,208,208,038,17
3,249,203,041,015,170,232,132,073,2427
1870 DATA 160,255,200,185,127,194,016,25
0,202,208,247,200,185,127,194,048,2798
1880 DATA 006,032,071,171,076,123,197,07
6,239,166,076,243,166,173,249,203,2267
1890 DATA 076,036,167,032,158,183,224,00
8,144,003,076,106,195,142,240,203,1993
1900 DATA 076,174,167,208,037,162,032,16
9,224,133,255,160,000,132,254,120,2303
1910 DATA 169,052,133,001,177,254,073,25
5,145,254,136,208,247,230,255,202,2791
1920 DATA 208,242,169,055,133,001,088,07
6,174,167,076,109,195,032,138,173,2036
1930 DATA 032,247,183,166,020,240,008,03
2,236,197,198,020,076,211,197,166,2229
1940 DATA 021,240,006,198,021,198,020,20
8,234,076,174,167,173,240,203,240,2419
1950 DATA 029,201,002,240,034,201,004,24
0,039,201,006,240,044,201,001,240,1923
1960 DATA 010,201,003,240,015,201,005,24
0,020,208,027,032,047,198,032,093,1572
1970 DATA 198,076,118,195,032,067,198,03
2,047,198,076,118,195,032,056,198,1836
1980 DATA 032,067,198,076,118,195,032,09
3,198,032,056,198,076,118,195,173,1857
1990 DATA 246,203,240,067,206,246,203,09
6,173,246,203,201,199,176,056,238,2999
2000 DATA 246,203,096,173,248,203,208,00
9,173,247,203,240,042,206,247,203,2947
2010 DATA 096,173,247,203,208,003,206,24
8,203,206,247,203,096,173,248,203,2963
2020 DATA 208,009,238,247,203,208,003,23
8,248,203,096,174,247,203,232,224,2981
2030 DATA 064,176,004,142,247,203,096,17
3,237,203,208,003,076,106,195,096,2229
2040 DATA 032,158,183,138,024,109,240,20
3,041,007,141,240,203,076,174,167,2136
2050 DATA 032,158,183,142,249,203,173,24
0,203,056,237,249,203,041,007,141,2517
2060 DATA 240,203,076,174,167,032,158,18
3,224,016,176,048,142,253,203,032,2327
2070 DATA 223,193,032,121,000,240,034,03
2,253,174,032,158,183,224,016,176,2091
2080 DATA 027,142,254,203,032,223,193,03
```

Listing Turtle-Grafik (Fortsetzung)

```
2,121,000,240,013,032,253,174,032,1971
2090 DATA 158,183,224,016,176,006,142,03
2,208,076,174,167,076,106,195,120,2059
2100 DATA 169,052,133,001,162,032,160,00
0,132,254,132,166,169,224,133,255,2174
2110 DATA 169,160,133,167,177,254,145,16
6,136,208,249,230,255,230,167,202,3048
2120 DATA 208,242,169,055,133,001,088,03
2,212,225,162,000,160,192,134,254,2267
2130 DATA 169,160,133,255,169,054,133,00
1,169,254,032,216,255,169,055,133,2357
2140 DATA 001,076,174,167,032,212,225,16
9,000,133,185,162,000,160,224,032,1952
2150 DATA 213,255,076,174,167,208,026,16
0,000,185,128,194,240,016,016,007,2065
2160 DATA 041,127,032,210,255,169,013,03
2,210,255,200,076,057,199,076,174,2126
2170 DATA 167,076,109,195,032,158,183,22
4,000,240,006,032,254,193,076,174,2119
2180 DATA 167,032,017,194,076,174,167,07
2,165,154,201,003,208,008,104,201,1943
2190 DATA 135,240,006,076,022,231,076,21
3,241,142,249,203,166,212,208,027,2447
2200 DATA 166,216,208,023,152,072,169,02
1,133,214,169,000,133,211,032,108,2027
2210 DATA 229,104,168,174,249,203,169,13
5,024,088,096,174,249,203,076,022,2363
2220 DATA 231,032,158,183,232,142,239,20
3,169,001,141,237,203,165,203,201,2740
2230 DATA 003,208,008,169,000,141,237,20
3,076,174,167,169,000,141,238,203,2137
2240 DATA 173,000,220,041,001,208,009,03
2,047,198,032,034,200,238,238,203,1874
2250 DATA 173,000,220,041,002,208,009,03
2,056,198,032,034,200,238,238,203,1884
2260 DATA 173,000,220,041,004,208,009,03
2,067,198,032,034,200,238,238,203,1897
2270 DATA 173,000,220,041,008,208,009,03
2,093,198,032,034,200,238,238,203,1927
2280 DATA 173,238,203,208,168,032,011,20
0,076,173,199,173,241,203,072,169,2539
2290 DATA 002,141,241,203,032,118,195,03
2,047,200,032,118,195,104,141,241,2042
2300 DATA 203,096,173,000,220,041,016,20
8,226,032,118,195,076,047,200,172,2023
2310 DATA 239,203,162,100,202,208,253,13
6,208,248,096,240,003,076,109,195,2678
2320 DATA 165,203,201,003,208,003,076,17
4,167,173,020,208,056,233,050,144,2084
2330 DATA 239,201,200,176,235,141,246,20
3,173,019,208,056,233,035,144,224,2733
2340 DATA 010,141,247,203,144,007,201,06
4,176,214,169,001,044,169,000,141,1931
2350 DATA 248,203,173,141,002,041,004,24
0,012,032,118,195,032,093,198,032,1764
2360 DATA 118,195,076,064,200,173,241,20
3,072,169,002,141,241,203,032,118,2248
2370 DATA 195,032,118,195,104,141,241,20
3,076,064,200,255,255,255,255,249,2838
63000 DATA -100:RETURN
READY.
```

Listing Turtle-Grafik (Schluß)

```
0 rem t u r t l e   d e m o
1 rem by peter menke
2 :
3 hires 1,0,0:color15:clear
4 :
10 rem roboter-kopf
20 plot 120,160
30 deg 2:move 10
40 deg 0:move 30
50 deg 2:move 10
55 deg 4:move 40
60 deg 2:move 100
65 deg 0:move 100
70 deg 6:move 100
90 deg 4:move 40
100 deg 6:move 10
110 deg 0:move 30
120 deg 6:move 10
121 :
122 rem linkes ohr
130 plot 110,105
140 deg 4:move 15
150 deg 2:move 30
160 deg 0:move 15
161 :
162 rem rechtes ohr
170 plot 210,105
180 deg 0:move 15
190 deg 2:move 30
200 deg 4;move 15
201 :
202 rem mund
210 plot 130,120
220 deg 6:move 10
230 deg 0:move 60
240 deg 2:move 10
241 :
242 rem nase
250 plot 152,100
260 deg 0:move 16
270 deg 3:move 8
280 deg 4:move 1
290 deg 5:move 8
291 :
292 rem linkes auge
293 plot 130,60:deg 0
```

Demo-Programm zur Turtle-Grafik

```
295 fort=1to4
300 move 15
310 rturn2
320 next
330 :
340 rem rechtes auge
350 plot 190,60:deg 6
360 fort=1to4
370 move 15
380 rturn 2
390 next
400 :
410 window 1:print"▒▒          t u r t l
e   d e m o"
420 print"▒              by peter menke"
430 fori=0to4000:next
440 window 0
450 :
460 :
461 rem spirale
470 hires 1,2,2:color 0:clear:plot 160,1
00
480 fori=1to66
490 lturn 1
500 move i
510 next
511 fori=1to1000:next
520 :
530 :
540 rem viereck-spiralen
541 clear:color1
550 hires 1,2,2:fori=1to200step2
560 rturn 2:move i:next
561 plot 160,100
570 fori=1to195step2
580 lturn 2:move i:next
590 plot 160,100
600 fori=1to195
610 rturn 2:move i:next
620 :
630 :
640 rem muster
641 hires1,5,5:clear:color0
650 fori=1to45
660 forx=0to7
670 rturnx
671 move i
680 nextx
700 next
710 :
720 hires1,5,5
730 fori=1to45
740 forx=0to7
750 lturnx
761 movei
780 nextx
790 next
800 fori=1to2000:next
810 :
820 :
830 rem pyramiden
840 hires1,6,6:clear:color15
850 fori=1to33
860 forx=0to7
870 deg x
871 move i
880 nextx
900 next
910 :
930 fori=1to33
940 forx=0to7
950 deg 7-x
961 move i
980 nextx
990 next
991 fori=1to8:revers
992 forx=1to500:next
993 next
1000 :
1010 :
1011 rem inverses muster
1020 hires7,7:color 0:clear
1030 fory=1to4:hires 1:mode 2
1040 fori=1to100
1050 move i
1060 forx=1to7
1070 lturn x:movex:nextx,i,y
1080 :
1090 :
1111 rem strich-muster
1120 hires1,8,8:color 0:clear
1130 fory=1to4:hires 1:mode2
1140 fori=1to45
1150 move i
1160 forx=1to10
1170 rturn x:move x:nextx,i,y
1181 :
1182 :
1183 rem joystick zeichnen
1184 hires 1,10,10:color0:clear
1185 window1
1186 print"▒▒     joystick-zeichnen (port
 2)"
1187 print"    weiter mit f7"
1188 joystick5
1189 :
1190 :
1191 rem lightpen zeichnen
1192 hires 1,1,1:color0:clear
1193 window1
1194 print"▒▒      lightpen-zeichnen (port
 1)"
1195 print"    weiter mit f7"
1196 lpen
1197 :
1198 :
1200 rem zufall
1210 hires 1:clear:color 1,9,9
1220 window 1
1230 print"▒▒zufallsmuster
1240 print"schluss mit run/stop-taste
1250 deg rnd(1)*8:move 1:goto1250

ready.
```

Demo-Programm zur Turtle-Grafik (Schluß)

Ohne gutes Werkzeug geht es nicht: SMON

TEIL 1

In mehreren Teilen möchten wir Ihnen einen Maschinensprachmonitor vorstellen. Parallel zum Kursus über Assembler-Programmierung wird Schritt für Schritt ein Programm entstehen, das sich durchaus mit kommerziellen Monitoren messen kann.

Ich kann mich noch gut an unsere ersten Schritte in die Maschinensprache erinnern. Ausgerüstet mit einer Befehlsliste für den 6502 und einem in Basic geschriebenen »Mini-Monitor« entstanden Programme, die 3 und 5 addieren und das Ergebnis im Speicher ablegen konnten. Dazu mußten wir die Befehlcodes aus der Liste heraussuchen und dann in den Speicher »POKEn«. Jeder Sprung mußte von Hand ausgerechnet werden, jeder falsch herausgesuchte Befehl führte zum Programmabsturz. Der erste Disassembler — ein Programm zur Anzeige der Maschinenbefehle in Assemblersprache — war für uns die Offenbarung. Von nun an konnten wir Maschinenprogramme analysieren und daraus lernen. Zum Verständnis der Maschinensprache ist es nämlich noch weit mehr als bei anderen Sprachen wichtig, vorhandene Programme zu verstehen und sich dabei die wichtigsten Techniken anzueignen.

Mit der Zeit wuchsen unsere Ansprüche, ein Assembler mußte her, um die neugewonnenen Erkenntnisse auch auszuprobieren. Das war zuerst wieder ein Basic-Programm, langsam und wenig komfortabel, aber immerhin. Wir schrieben unsere ersten kleinen Routinen, vor allem, um vorhandene Maschinenprogramme unseren eigenen Wünschen anzupassen. Mit dem AMON für den VC 20 bekamen wir dann einen Monitor, der (fast) alle unsere Wünsche erfüllte. Als wir jedoch auf den C 64 umstiegen, mußten wir feststellen, daß es für diesen Computer nichts gab, das uns zufriedenstellen konnte. Der einzige Ausweg: Selbst programmieren. So entstand im Laufe eines Jahres SMON. Ursprünglich hatten wir nur vor, die Funktionen von AMON für den C 64 zu programmieren, aber dabei blieb es nicht. Immer neue Befehle und Routinen kamen hinzu, bis wir endlich zufrieden waren.

Was bietet SMON?

Zunächst ist alles enthalten, was zum »Standard« gehört: Memory-Dump, also die Anzeige des Speicherinhalts in Hexbytes, mit Änderungsmöglichkeiten, ein Disassembler mit Änderungsmöglichkeit sowie Routinen zum Laden, Abspeichern und Starten von Maschinenprogrammen. Darüber hinaus gibt es einen kleinen Direktassembler, der sogar Labels verarbeitet, Befehle zum Verschieben im Speicher mit und ohne Umrechnen der Adressen und Routinen zum Umrechnen von Hex-, Dezimal- und Binärzahlen. Der besondere Clou von SMON liegt aber zweifellos in seinen leistungsfähigen Suchroutinen und vor allem im Trace-Modus. Damit lassen sich Maschinenprogramme Schritt für Schritt abarbeiten und kontrollieren.

Dieser erste Teil umfaßt sämtliche Eingabe- und Ausgaberoutinen, die Registeranzeige, den Memory-Dump sowie Disassembler und Assembler. Damit steht Ihnen bereits ein lauffähiges Monitorprogramm mit den unten aufgeführten Befehlen zur Verfügung.

Der Monitor benötigt für alle Eingaben die hexadezimale Schreibweise, das heißt zu den Zahlen 1 bis 9 kommen noch die Buchstaben A (für dez. 10) bis F (für dez. 15) hinzu.

Bei der Eingabe von Adressen ist folgendes zu beachten: [ANFADR] bedeutet exakt die Startadresse, [ENDADR] bedeutet hierbei die erste Adresse hinter dem gewählten Bereich. Im Normalfall ist die Eingabe mit und ohne Leerzeichen zulässig. Beim Abweichen von dieser Regel wird darauf besonders verwiesen.

Assemblieren

A [ANFADR]

Assemblierung beginnt bei angegebener Adresse

Beispiel:

A 4000 Beginn bei Startadresse $4000

Nach Eingabe von »RETURN« erscheint auf dem Bildschirm die gewählte Adresse mit einem blinkenden Cursor. Die Befehle werden so eingegeben, wie sie der Disassembler zeigt: LDY #00 oder LDA 400E,Y und so weiter. »RETURN« schließt die Eingabe der Zeile ab. Bei fehlerhafter Eingabe springt der Cursor wieder in die Anfangsposition zurück. Ansonsten wird der Befehl disassembliert und nach Ausgabe der Hex-Bytes gelistet. Zur Korrektur vorhergehender Zeilen gehen Sie mit dem Cursor zur Anfangsposition (hinter die Adresse) zurück, schreiben den Befehl neu und gehen nach »RETURN« mit dem Cursor wieder in die letzte Zeile. Falls Ihnen bei Sprüngen (Branch-Befehl, JSR und JMP) die Zieladressen noch nicht bekannt sind, geben Sie einfach sogenannte »Label« ein.

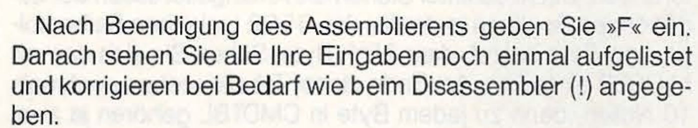
Ein Label besteht aus dem Buchstaben »M« (für Marke) und einer zweistelligen Hex-Zahl von 01 bis 30.

Zum Beispiel: BCC M01

Wenn Sie die Zieladresse für diesen Sprung erreicht haben, dann kennzeichnen Sie diese mit eben dieser »Marke«.

Zum Beispiel: M01 LDY #00

Einzelne Bytes nimmt der Assembler an, indem Sie diese mit einem Punkt kennzeichnen: .00 oder .AB. In diesem Modus werden die Eingaben natürlich nicht disassembliert.



Nach Beendigung des Assemblierens geben Sie »F« ein. Danach sehen Sie alle Ihre Eingaben noch einmal aufgelistet und korrigieren bei Bedarf wie beim Disassembler (!) angegeben.

Probieren Sie einmal das folgende Beispiel:
A 4000

Der Assembler meldet sich mit: »4000« und einem blinkenden Cursor. Geben Sie nun ein (die Adressen erscheinen automatisch):

```
4000 LDY #00        4009 CPY #12
4002 LDA 400E,Y     400B BCC 4002
4005 JSR FFD2       400D BRK
4008 INY
```

Die folgenden Bytes werden wie beschrieben mit einem Punkt eingegeben. Sie werden nicht disassembliert.

```
400E .0D            4017 .54
400F .0D            4018 .20
4010 .53            4019 .53
4011 .4D            401A .55
4012 .4F            401B .50
4013 .4E            401C .45
4014 .20            401D .52
4015 .49            401E .0D
4016 .53            401F .0D
```

Drücken Sie anschließend »F«. Ihr Programm wird nochmal aufgelistet. Starten Sie es nun mit »G 4000«. Es erscheint ein Text auf dem Bildschirm — lassen Sie sich überraschen.

Disassemblieren

D [ANFADR,ENDADR]
disassembliert den Bereich von ANFADR bis ENDADR, wobei ENDADR nicht eingegeben werden muß. Wird keine Endadresse eingegeben, erscheint zunächst nur eine Zeile:

```
ADR     HEXBYTES   BEFEHL
4000    A000       LDY #00
```

Mit der SPACE-Taste wird der jeweils nächste Befehl in der gleichen Art und Weise gezeigt. Wünschen Sie eine fortlaufende Ausgabe, drücken Sie »RETURN«. Die Ausgabe wird dann so lange fortgesetzt, bis eine weitere Taste gedrückt wird oder bis ENDADR erreicht ist. Mit »RUN/STOP« springen Sie jederzeit in den Eingabemodus zurück.

Das Komma, das vor der Adresse auf dem Bildschirm erscheint, ist ein »hidden command« (verstecktes Kommando). Es braucht nicht eingegeben zu werden, da es automatisch beim Disassemblieren angezeigt wird. So ermöglicht es ein einfaches Ändern des Programms. Fahren Sie mit dem Cursor auf den zu ändernden Befehl und überschreiben Sie ihn mit dem neuen. Wenn Sie jetzt »RETURN« drücken, erkennt SMON das Komma als Befehl und führt ihn im Speicher aus. Achten Sie aber darauf, daß der neue Befehl die gleiche Länge (in Bytes) hat und füllen Sie gegebenenfalls mit »NOPs« auf. Zur Kontrolle können Sie den geänderten Bereich noch einmal disassemblieren.

Lassen Sie als Beispiel einmal das Programm (siehe Befehl »A«) ab 4000 disassemblieren (»D 4000 4011«). Ändern Sie nun den ersten Befehl auf LDY #01. Die Änderung zeigt sich daran, daß die HEX-Bytes automatisch den neuen Wert annehmen. Starten Sie nun das Programm nochmals mit »G 4000«. Jetzt erscheint der Text mit nur einer Zeile Abstand auf dem Bildschirm.

Starten eines Maschinenprogramms (Go)

G [ADRESSE]
startet ein Maschinenprogramm, das bei ADRESSE beginnt. Das Programm muß mit einem BRK-Befehl abgeschlossen werden, damit ein Rücksprung in SMON erfolgen kann. Wird nach »G« keine Adresse eingegeben, benutzt SMON die, die mit dem letzten BRK erreicht worden ist und bei der Register-Ausgabe als PC auftaucht. Mit dem »R«-Befehl (siehe unten) werden die Register vorher auf gewünschte Werte gesetzt.

Memory-Dump

M [ANFADR ENDADR]
gibt die HEX-Werte des Speichers sowie die zugehörigen ASCII-Zeichen aus. Auch hier kann auf die Eingabe einer Endadresse verzichtet werden. Die Steuerung der Ausgabe entspricht der beim Disassemblieren.

Beispiel:
M 4000 gibt die Inhalte der Speicherstellen $4000 bis $4007 aus. Weiter geht es wie beim Disassemblieren mit SPACE oder RETURN. Die Bytes können ebenfalls durch Überschreiben geändert werden, allerdings nicht die ASCII-Zeichen. Verantwortlich dafür ist der Doppelpunkt, der am Anfang jeder Zeile ausgegeben wird, ein weiterer »hidden command«. Wenn Ihre Änderung nicht durchgeführt werden kann, weil Sie zum Beispiel versuchen, ins ROM zu schreiben, wird ein »?« als Fehlermeldung ausgegeben.

Registeranzeige

R zeigt den gegenwärtigen Stand der wichtigsten 6510-Register an: Programmzähler (PC), Status-Register (SR), Akkumulator (AC), X-Register (XR), Y-Register (YR), Stackpointer (SP). Außerdem werden die einzelnen Flags des Status-Registers mit 1 für »gesetzt« und 0 für »nicht gesetzt« angezeigt. Durch Überschreiben werden die Inhalte auf einen gewünschten Wert gesetzt. Die Flags können allerdings nicht einzeln verändert werden, sondern nur durch Überschreiben des Wertes von SR.

Exit

X springt ins Basic zurück. Alle Basic-Pointer bleiben erhalten. Sie können also zum Beispiel direkt im Programm fortfahren, wenn Sie zwischendurch mit SMON einige Speicherstellen kontrolliert haben.

Probieren Sie alle bisher beschriebenen Befehle in Ruhe aus und machen Sie sich mit SMON vertraut. Arbeiten Sie auch parallel den Kurs über Assemblerprogrammierung in dieser Ausgabe durch. Alle Beispiele dort sind auf SMON abgestimmt.

Wir wollen jetzt einen Blick auf das Programm selbst werfen. Natürlich ist es unmöglich, den gesamten Quelltext umfassend zu beschreiben. Andererseits enthält SMON aber eine Reihe von Routinen, die in jedem Maschinenprogramm vorkommen. Wir werden im Rahmen dieser Serie versuchen, die wichtigsten zu erklären, damit Sie sie später in eigene Programme einbauen können.

Zum besseren Verständnis werden solche Routinen so abgedruckt, wie wir sie im Assembler-Quelltext geschrieben haben. Sie enthalten daher anstelle absoluter Adressen Labels, deren Name — hoffentlich — etwas über den Sinn und Zweck aussagt. Parallel dazu sollten Sie sich diese Routinen von SMON disassemblieren lassen, damit Sie sehen, wie es denn nun fertig im Speicher aussieht.

Beginnen wir mit der Routine GETCHRERR. Das soll soviel bedeuten wie »Hole ein Zeichen und erzeuge eine Fehlermeldung, wenn keins eingegeben wurde«. Leider wäre so ein Label auch für den geduldigsten Assembler viel zu lang, daher die merkwürdige Abkürzung. Mit dieser Routine holen wir ein Zeichen von der Tastatur. Das erledigt die Betriebssystemroutine CHRIN. Um zu prüfen, ob überhaupt etwas eingegeben wurde, untersuchen wir das Zeichen. Handelt es sich um die »RETURN«-Taste ($0D), hat der Benutzer gar kein Zeichen eingegeben. Dies quittiert SMON mit einem »?« und dem Rücksprung in den Eingabemodus. So läßt sich — in gewissen Grenzen — kontrollieren, ob zu einem Befehl die richtigen Eingaben gemacht wurden. Geben Sie einmal den »D«-Befehl ohne Angabe einer Adresse ein, dann sehen Sie, was gemeint ist.

Alle Eingaberoutinen benutzen GETCHRERR, um Falscheingaben zu prüfen. Nehmen wir als Beispiel GETBYT. Diese soll ein Byte, also zwei ASCII-Zeichen 0 - F von der Tastatur holen und in ein Byte umwandeln. Das erste Zeichen wird darauf überprüft, ob es sich um ein »Space« oder ein Komma handelt. Trifft das zu, wird es einfach übergangen und das nächste Zeichen geholt. Der Benutzer kann also Leerzeichen und Komma benutzen, um seine Eingaben übersichtlicher zu machen, er muß aber nicht! Ist das Zeichen aber gültig, wird es von ASCHEX in eine Hexzahl gewandelt.

Dazu ein Beispiel:

Auf der Tastatur wurde 5B eingetippt. Zuerst wird jetzt die 5 (ASCII $35) mit $3A verglichen, um festzustellen, ob es sich um eine Zahl (0 - 9) oder einen Buchstaben (A - F) handelt. ASCII $35 ist eine Zahl, also wird nur die linke Hälfte ausmaskiert (AND #$0F). Ergebnis ist $05. Jetzt wird viermal nach links geschoben und das Ergebnis ($50) in $B4 zwischengespeichert. Nun ist das B (ASCII $42) an der Reihe. Da $42 größer ist als $3A werden diesmal 8 und das gesetzte Carry-Flag, also 9 addiert. Ergebnis ist $5B. Linke Hälfte ausmaskieren wie gehabt und eine OR-Verknüpfung mit dem gemerkten $50 ergibt $5B. Das war's.

Meistens aber braucht SMON zwei Bytes als Eingabe, zum Beispiel für Adressen. Mit dem, was wir schon haben, kein Problem: GETADR ruft einfach GETBYT zweimal hintereinander auf und legt das Ergebnis in zwei Speicherstellen in der Zeropage ab, die mit dem X-Register ausgewählt werden können. Brauchen wir mehr als eine Adreßeingabe, rufen wir einfach GETADR mehrmals auf. So etwas machen GET3ADR und GET2ADR. Bisweilen aber, zum Beispiel beim G-Befehl, darf eine Adresse eingegeben werden, es muß aber nicht sein. Deswegen prüft GETSTART, ob direkt nach dem »G« »RETURN« gedrückt wurde. Dies erledigt GETRET. Wenn ja, wird die Adresse benutzt, die in PCL und PCH steht. Das sind SMONs interne Programm-Counter. Ansonsten wird die eingetippte Adresse benutzt.

Sie sehen, wie aus einfachen Routinen immer kompliziertere Befehle zusammengesetzt werden. Und das ist das ganze Geheimnis, wenn Sie umfangreiche Programme schreiben: Gliedern Sie sich das Problem (hier eine benutzerfreundliche Eingabe) in kleine und kleinste Schritte auf, die Sie dann jeden für sich programmieren und austesten.

Werfen wir noch einen Blick auf die Art und Weise, wie SMON Befehle verarbeitet. In EXECUTE setzen wir zunächst den Stackpointer auf den Wert, den er beim letzten BRK erreicht hatte. Dann werden als erstes die »hidden commands« abgeprüft. Wir lesen dazu direkt vom Bildschirm. D3 enthält die Anfangsadresse der aktuellen Zeile im Speicher. Übrigens gibt es neben den bereits erwähnten noch weitere »hidden commands«, die in den späteren Folgen noch auftauchen werden. Liegt kein verstecktes Kommando vor, holen wir mit GETCHRERR ein Zeichen und merken es uns in COMMAND. Jetzt untersuchen wir, ob dieses Zeichen in der Befehlsliste (CMDTBL) steht. CMDTBL steht übrigens ab $C00B ganz oben im Speicher. Sie endet mit fünf Nullen für spätere Erweiterungen. Direkt dahinter stehen die Anfangsadressen der zugehörigen Routinen in der für den 6502 typischen Reihenfolge, Low-Byte zuerst, dann High-Byte. Sehen Sie sich das mit M C00B einmal an. Am Ende dieser Tabelle stehen nochmals 10 Nullen, denn zu jedem Byte in CMDTBL gehören ja zwei Adreßbytes in der Liste (CMDS). Wenn nun ein Kommando in CMDSEARCH gefunden wurde, wird CMDEXEC als Subroutine aufgerufen. CMDEXEC legt nun die zugehörigen Adreßbytes auf den Stack und führt dann einen RTS aus, der jetzt — nach der Stackmanipulation — zu dem gewünschten Befehl führt. Beachten Sie, daß RTS immer auf die um eins erhöhte Adresse springt, daher müssen Sie zu den Adressen in CMDS immer 1 addieren, wenn Sie den Anfang einer Routine suchen.

Alle Befehle in SMON enden mit einem RTS, springen also auf den JMP EXECUTE hinter CMDFOUND. Damit ist eine Endlosschleife geschlossen, die immer einen Befehl ausführt und anschließend wieder in die Eingabe zurückspringt. Beim nächsten Mal erfahren Sie etwas über LOAD, SAVE und die Umrechnung verschiedener Zahlensysteme.

(Dietrich Weineck/N. Mann/gk)

Hinweise zum Abtippen

Es ist mal wieder eine DATA-Wüste, die wir Ihnen zumuten, aber wenn Sie die erfolgreich hinter sich brachten, haben Sie schon mehr als die Hälfte vom gesamten SMON geschafft. Um Ihnen das Abtippen beziehungsweise die anschließende — fast unvermeidliche — Fehlersuche so einfach wie möglich zu machen, unterteilten wir das Gesamtprogramm in Blöcke zu je 256 Bytes, die jeweils eine eigene Prüfsumme haben. Wenn Sie sich vertippt haben, erscheint eine Fehlermeldung mit Angabe des Blocks, in dem sich der Fehler — höchstwahrscheinlich — befindet.

Vor dem ersten »RUN« sollten Sie aber unbedingt das Programm speichern, sonst kann Ihnen bei Fehlern der Computer abstürzen, und alle Mühe war umsonst.

Eins findet die Prüfsummenmethode allerdings nicht, nämlich zuviel eingegebene Nullen oder Kommas. Erhalten Sie aber keine Fehlermeldung und das Programm läuft trotzdem nicht, kontrollieren Sie als erstes, ob wirklich alle DATAs »aufgebraucht« sind. Dazu tippen Sie im Direktmodus PRINT A ein. Jetzt muß die letzte Zahl, also 197 erscheinen. Wenn nicht, haben Sie eine 0 oder ein Komma zu viel.

Wenn das Ladeprogramm endlich ohne Fehler bis zum READY durchläuft, können Sie SMON mit SYS 49152 starten. Die Bildschirmfarben ändern sich und es erscheint die Registeranzeige und in der nächsten Zeile ein Punkt mit blinkendem Cursor. Probieren Sie jetzt alle Kommandos durch. Hüten Sie sich aber vor allen anderen Kommandos. Die Fehlermeldung bei falschen Kommandos funktioniert noch nicht richtig!! Deshalb führen Falscheingaben in den meisten Fällen zum Programmabsturz. Das wird sich im Verlauf dieser Serie allerdings noch ändern.

Bevor Sie Ihren Computer aus dem Fenster werfen, noch ein Hinweis: Von der nächsten Folge ab wird SMON auch fix und fertig im Leserservice zu erhalten sein.

Und noch ein letzter Tip: Das Wichtigste, was ein angehender Maschinenprogrammierer braucht, ist ein Reset-Taster. (Bauanleitungen oder fertige Taster wurden schon oft im 64'er vorgestellt.) Sie werden es merken, wenn Sie mit sorgenzerfurchter Stirn, den Tränen nahe, vor Ihrem Bildschirm sitzen, kein freundlich blinkender Cursor weit und breit und RUN/STOP RESTORE auch dann nichts mehr bringt, wenn Sie die Tasten durch das Gehäuse durchdrücken. Verzweifeln Sie nicht, drücken Sie RESET, starten Sie SMON neu mit SYS 49152 und schon können Sie bis zum nächsten Absturz weiterarbeiten......

Listing 1. Der DATA-Lader für SMON — Teil 1

```
10 REM **************************
20 REM *                        *
30 REM *     SMON TEIL 1        *
40 REM * VON N.MANN & D.WEINECK *
50 REM *     FLEETRADE 40       *
60 REM *     2800 BREMEN        *
70 REM *   TEL. 0421 / 493090   *
80 REM *                        *
90 REM **************************
100 FORI=0TO8:READA:PR(I)=A:NEXT
110 SA=49152:I=0
120 PA=SA+256*I:CH=0
130 FORJ=0TO255:READA:POKEPA+J,A:CH=CH+A
:NEXT
140 IFCH<>PR(I)THEN190
150 I=I+1:IFI<8THEN120
160 PA=PA+256:CH=0
170 FORJ=0TO60:READA:POKEPA+J,A:CH=CH+A:
NEXT
180 IFCH=PR(I)THENEND
190 PRINT"FEHLER IN BLOCK"I+1:END
191 REM
192 REM *** BLOCKPRUEFSUMMEN ***
195 DATA20921,25604,31944,33700,36302,34
378,34305,34639,7819
200 REM
210 REM *** BLOCK 1 ***
220 REM
230 DATA169,20,141,22,3,169,194,141,23,3
,0,39,35,36,37,44,58,59,61,63,65,66
240 DATA67,68,70,71,73,75,76,77,79,80,82
,83,84,86,87,88,0,0,0,0,0,218,202,45
250 DATA201,7,201,27,201,251,198,28,196,
181,195,244,202,153,200,208,198,107
260 DATA201,60,202,92,197,16,203,226,195
,67,200,182,202,77,200,248,195,192
270 DATA201,60,200,133,195,77,200,240,20
3,66,202,210,201,109,195,0,0,0,0,0
280 DATA0,0,0,0,0,255,255,1,0,65,90,73,8
2,84,128,32,64,16,0,2,1,1,2,0,145,145
290 DATA13,83,217,49,55,50,13,0,125,76,1
25,201,13,13,32,32,80,67,32,32,83,82
300 DATA32,65,67,32,88,82,32,89,82,32,83
,80,32,32,78,86,45,66,68,73,90,67,0
310 DATA2,4,1,44,0,44,89,41,88,157,31,25
5,28,28,31,31,31,28,223,28,31,223,255
320 DATA255,3,31,128,9,32,12,4,16,1,17,2
0,150,28,25,148,190,108,3,19,1,2,2
330 DATA3,3,2,2,2,2,2,2,3,3,2,3,3,3,2,0,
64,64,128,128,32,16,37,38,33,34,129
340 DATA130,33,130,132,8,8,231,231,231,2
31
350 REM
360 REM *** BLOCK 2 ***
370 REM
380 DATA227,227,227,227,227,227,227,227,
227,227,231,167,231,231,243,243,247
390 DATA223,38,70,6,102,65,129,225,1,160
,162,161,193,33,97,132,134,230,198
400 DATA224,192,36,76,32,144,176,240,48,
208,16,80,112,120,0,24,216,88,184,202
410 DATA136,232,200,234,72,8,104,40,64,9
6,170,168,186,138,154,152,56,248,137
420 DATA156,158,178,42,74,10,106,79,35,1
47,179,243,51,211,19,83,115,82,76,65
430 DATA82,69,83,83,79,76,76,76,67,65,65
,83,83,73,68,67,67,66,74,74,66,66,66
440 DATA66,66,66,66,66,83,66,67,67,67,67
,68,68,73,73,78,80,80,80,80,82,82,84
450 DATA84,84,84,84,84,83,83,79,83,83,79
,79,84,66,82,68,68,68,77,78,68,84,84
460 DATA78,69,80,80,73,77,83,67,67,69,77
,78,80,86,86,69,82,76,76,76,76,69,69
470 DATA78,78,79,72,72,76,76,84,84,65,65
,83,88,88,89,69,69,76,82,76,82,82,65
480 DATA67,65,89,88,65,80,68,67,89,88,67
,67,88,89,84,80,82,67,83,81,73,69,76
490 DATA67,83,73,75,67,68,73,86,88,89,88
,89,80,65,80,65,80,73,83,88,89,88,65
500 REM
510 REM *** BLOCK 3 ***
520 REM
530 DATA83,65,67,68,8,132,129,34,33,38,3
2,128,3,32,28,20,20,16,4,12,216,169
540 DATA8,141,176,2,169,4,141,175,2,169,
6,141,32,208,141,33,208,169,3,141,134
550 DATA2,162,5,104,157,168,2,202,16,249
,173,169,2,208,3,206,168,2,206,169
560 DATA2,186,142,174,2,169,82,76,255,19
4,32,194,194,240,11,32,126,194,141
570 DATA169,2,165,252,141,168,2,96,162,1
64,32,128,194,32,128,194,208,28,32
580 DATA126,194,169,254,133,253,169,255,
133,254,32,194,194,208,12,141,119,2
590 DATA230,198,96,32,126,194,44,162,251
,32,141,194,149,1,32,154,194,149,0
600 DATA232,232,96,32,202,194,201,32,240
,249,201,44,240,245,208,3,32,202,194
610 DATA32,175,194,10,10,10,10,133,180,3
2,202,194,32,175,194,5,180,96,201,58
620 DATA144,2,105,8,41,15,96,32,202,194,
201,32,240,249,198,211,96,32,207,255
630 DATA198,211,201,13,96,32,207,255,201
,13,208,248,169,63,32,210,255,174,174
640 DATA2,154,162,0,134,198,32,81,195,16
1,209,201,39,240,17,201,58,240,13,201
650 DATA59,240,9,201,44,240,5,169,46,32,
210,255,32,202,194,201,46,240,249,133
660 REM
670 REM *** BLOCK 4 ***
680 REM
690 DATA172,41,127,162,32,221,10,192,240
,5,202,208,248,240,194,32,21,195,76
700 DATA214,194,138,10,170,232,189,41,19
2,72,202,189,41,192,72,96,165,252,32
710 DATA42,195,165,251,72,74,74,74,74,32
,53,195,104,41,15,201,10,144,2,105
720 DATA6,105,48,76,210,255,169,13,32,21
0,255,138,76,210,255,32,76,195,169
730 DATA32,76,210,255,169,13,76,210,255,
133,187,132,188,160,0,177,187,240,6
740 DATA32,210,255,200,208,246,96,230,25
1,208,2,230,252,96,169,14,141,134,2
750 DATA141,32,208,169,6,141,33,208,169,
55,133,1,174,174,2,154,76,116,164,160
760 DATA192.169.140,32,86,195,162,59,32,
64,195,173,168,2,133,252,173,169,2
770 DATA133,251,32,35,195,32,76,195,162,
251,189,175,1,32,42,195,32,76,195,232
780 DATA208,244,173,170,2,76,208,195,32,
78,194,162,251,32,202,194,32,154,194
790 DATA157,175,1,232,208,244,32,76,195,
189,170,2,76,208,195,133,170,169,32
800 DATA160,9,32,210,255,6,170,169,48,10
5,0,136,208,244,96,32,73,194,174,174
```

```
810 DATA2,154,162,250,189,174,1,72,232,2
08,249,104,168,104,170,104,64,32,100
820 DATA194,162,58,32,64
830 REM
840 REM *** BLOCK 5 ***
850 REM
860 DATA195,32,35,195,160,32,162,0,32,76
,195,161,251,32,42,195,161,251,32,57
870 DATA196,208,241,32,93,196,144,224,96
,32,126,194,160,32,162,0,32,202,194
880 DATA32,154,194,129,251,193,251,240,3
,76,209,194,32,57,196,208,236,96,201
890 DATA32,144,12,201,96,144,10,201,192,
144,4,201,219,144,4,169,46,41,63,41
900 DATA127,145,209,173,134,2,145,243,32
,103,195,200,192,40,96,32,111,196,76
910 DATA102,196,32,103,195,165,251,197,2
53,165,252,229,254,96,32,148,196,32
920 DATA134,196,240,14,32,134,196,240,25
1,201,32,208,5,141,119,2,230,198,96
930 DATA32,228,255,72,32,225,255,240,2,1
04,96,76,214,194,160,40,36,172,16,246
940 DATA132,200,132,208,169,255,32,195,2
55,169,255,133,184,133,185,173,175
950 DATA2,133,186,32,192,255,162,0,134,2
11,202,32,201,255,32,207,255,32,210
960 DATA255,201,13,208,246,32,204,255,16
9,145,76,210,255,160,0,177,251,36,170
970 DATA48,2,80,12,162,31,221,60,193,240
,47,202,224,21,208,246,162,4,221,73
980 DATA193,240,33,221,77,193,240,30,202
,208,243,162,56,221,17,193,240,20,202
990 DATA224,22,208,246,177,251,61,251
1000 REM
1010 REM *** BLOCK 6 ***
1020 REM
1030 DATA192,93,17,193,240,5,202,208,243
,162,0,134,173,138,240,15,162,17,177
1040 DATA251,61,181,192,93,198,192,240,3
,202,208,243,189,234,192,133,171,189
1050 DATA216,192,133,182,166,173,96,160,
1,177,251,170,200,177,251,160,16,196
1060 DATA171,208,7,32,74,197,160,3,208,2
,164,182,142,174,0,141,175,0,96,160
1070 DATA1,177,251,16,1,136,56,101,251,1
70,232,240,1,136,152,101,252,96,162
1080 DATA0,134,170,32,100,194,32,140,197
,165,173,201,22,240,12,201,47,240,8
1090 DATA201,33,240,4,201,48,208,13,32,8
1,195,162,35,169,45,32,210,255,202,208
1100 DATA250,32,93,196,144,217,96,162,44
,32,64,195,32,35,195,32,76,195,32,117
1110 DATA198,32,203,196,32,76,195,177,25
1,32,42,195,32,76,195,200,196,182,208
1120 DATA243,169,3,56,229,182,170,240,9,
32,73,195,32,76,195,202,208,247,169
1130 DATA32,32,210,255,160,0,166,173,208
,17,162,3,169,42,32,210,255,202,208
1140 DATA248,36,170,48,133,76,106,198,36
,170,80,41,169,8,36,171,240,35,177,251
1150 DATA41,252,133,173,200,177,251,10,1
68,185,60,3,141,174,0,200,185,60,3,141
1160 DATA175,0,32,190,198,164
1170 REM
1180 REM *** BLOCK 7 ***
1190 REM
1200 DATA182,32,147,198,32,203,196,189,9
1,193,32,210,255,189,147,193,32,210
1210 DATA255,189,203,193,32,210,255,169,
32,36,171,240,3,32,73,195,162,32,169
1220 DATA4,36,171,240,2,162,40,138,32,21
0,255,36,171,80,5,169,35,32,210,255
1230 DATA32,44,197,136,240,22,169,8,36,1
71,240,7,169,77,32,210,255,160,1,185
1240 DATA173,0,32,42,195,136,208,247,160
,3,185,172,192,36,171,240,9,185,175
1250 DATA192,190,178,192,32,66,195,136,2
08,237,165,182,32,103,195,56,233,1,208
1260 DATA248,96,164,211,169,32,145,209,2
00,192,40,144,249,96,228,171,208,4,5
1270 DATA173,133,173,96,185,173,0,145,25
1,209,251,208,4,136,16,244,96,104,104
1280 DATA96,208,28,138,5,171,133,171,169
,4,133,181,32,207,255,201,32,240,13
1290 DATA201,36,240,9,201,40,240,5,201,4
4,240,1,96,198,181,208,232,96,224,24
1300 DATA48,14,173,174,0,56,233,2,56,229
,251,141,174,0,160,64,96,32,126,194
1310 DATA133,253,165,252,133,254,32,81,1
95,32,228,198,48,251,16,246,169,0,133
1320 DATA211,32,76,195,32,35,195,32,76,1
95,32,207,255,169,1,133,211,162,128
1330 DATA208,5,162,128,142,177
1340 REM
1350 REM *** BLOCK 8 ***
1360 REM
1370 DATA2,134,170,32,126,194,169,37,133
,200,44,177,2,16,8,162,10,32,207,255
1380 DATA202,208,250,169,0,141,177,2,32,
161,198,201,70,208,22,70,170,104,104
1390 DATA162,2,181,250,72,181,252,149,25
0,104,149,252,202,208,243,76,100,197
1400 DATA201,46,208,17,32,154,194,160,0,
145,251,209,251,208,4,32,103,195,200
1410 DATA136,96,162,253,201,77,208,25,32
,154,194,160,0,201,63,176,239,10,168
1420 DATA165,251,153,60,3,165,252,200,15
3,60,3,32,161,198,149,169,224,253,208
1430 DATA4,169,7,133,183,232,208,240,162
,56,165,166,221,91,193,240,5,202,208
1440 DATA246,202,96,165,167,221,147,193,
208,244,165,168,221,203,193,208,237
1450 DATA189,17,193,133,173,32,161,198,1
60,0,224,32,16,9,201,32,208,8,189,77
1460 DATA193,133,173,76,49,200,160,8,201
,77,240,32,160,64,201,35,240,26,32,157
1470 DATA194,141,174,0,141,175,0,32,161,
198,160,32,201,48,144,27,201,71,176
1480 DATA23,160,128,198,211,32,161,198,3
2,157,194,141,174,0,32,161,198,192,8
1490 DATA240,3,32,190,198,132,171,162,1,
201,88,32,154,198,162,4,201,41,32,154
1500 DATA198,162,2,201,89,32,154,198
1510 REM
1520 REM *** BLOCK 9 ***
1530 REM
1540 DATA165,173,41,13,240,10,162,64,169
,8,32,129,198,169,24,44,169,28,162,130
1550 DATA32,129,198,160,8,165,173,201,32
,240,9,190,3,194,185,11,194,32,129,198
1560 DATA136,208,244,165,171,16,1,200,20
0,32,138,198,198,183,165,183,133,211
1570 DATA76,151,197

READY.
```

Listing 1. Der DATA-Lader für SMON — Teil 1 (Schluß)

Listing 2. Der Assembler-Quelltext von SMON — Teil 1

```
;***************************
;*                         *
;*        S M O N          *
;*                         *
;* MASCHINENSPRACH-MONITOR *
;*                         *
;*        T E I L  1       *
;*                         *
;*       AUGUST 1984       *
;*                         *
;*  BY N.MANN & D.WEINECK  *
;*   TEL. 0421 / 493090    *
;*                         *
;***************************
;
;
                  .BA $C000
                  .OS
                  .CE
PCL               .DE $FB
PCH               .DE PCL+1
FLAG              .DE $AA
COMMAND           .DE $AC
;
BEFCODE           .DE $AD
ADRCODE           .DE $AB
BEFLEN            .DE $B6
LOPER             .DE $AE
HOPER             .DE LOPER+1
;
PCHSAVE           .DE $02A8
PCLSAVE           .DE PCHSAVE+1
SRSAVE            .DE PCHSAVE+2
AKSAVE            .DE PCHSAVE+3
XRSAVE            .DE PCHSAVE+4
YRSAVE            .DE PCHSAVE+5
SPSAVE            .DE PCHSAVE+6
;
PRINTNR           .DE PCHSAVE+7
IO.NR             .DE PCHSAVE+8
MEM               .DE PCHSAVE+9
;
TASTBUF           .DE $0277
COLOR             .DE $0286
BUF1              .DE $033C
BUF2              .DE BUF1+$30
BUF3              .DE BUF2+$30
BUF4              .DE BUF3+$30
;
READY             .DE $A474
BORDER            .DE $D020
BKGRND            .DE $D021
OPEN              .DE $FFC0
CLOSE             .DE $FFC3
CHKOUT            .DE $FFC9
CLRCHN            .DE $FFCC
CHRIN             .DE $FFCF
CHROUT            .DE $FFD2
STOPT             .DE $FFE1
GETIN             .DE $FFE4
;
;
SETBRK            LDA #L,BREAK
                  STA $0316
                  LDA #H,BREAK
                  STA $0317
                  BRK
;
CMDTBL            .BY $27
                  .BY '#$%,:;=?A'
                  .BY 'BCDFGIKL'
                  .BY 'MOPRSTVWX'
                  .BY $00 $00 $00 $00 $00
;
CMDS              .SE TICK-1
                  .SE BEFDEC-1
                  .SE BEFHEX-1
                  .SE BEFBIN-1
                  .SE KOMMA-1
                  .SE COLON-1
                  .SE SEMIS-1
                  .SE COMP-1
                  .SE ADDSUB-1
                  .SE ASSEMBLER-1
                  .SE BDATA-1
                  .SE CONVERT-1
                  .SE DISASSEM-1
                  .SE FINDE-1
                  .SE GO-1
                  .SE IO.SET-1
                  .SE KONTROLLE-1
                  .SE LOADSAVE-1
                  .SE MEMDUMP-1
                  .SE OCCUPY-1
                  .SE SETPRINTER-1
                  .SE REGISTER-1
                  .SE LOADSAVE-1
                  .SE TRACE-1
                  .SE VERSCHIEBE-1
                  .SE WRITE-1
                  .SE EXIT-1
                  .DS 10
;
OFFSET            .BY $FF $FF $01 $00
;
FINDTAB           .BY 'AZIRT'
FINDFLG           .BY $80 $20 $40 $10 $00
FINDFLG1          .BY $02 $01 $01 $02 $00
;
SYS172            .BY $91 $91 $0D $53
                  .BY $D9 $31 $37 $32 $0D
;
DATATAB           .BY $00 $7D $4C
                  .SE DATALOOP
;
REGHEAD           .BY $0D $0D $20 $20
                  .BY 'PC  SR AC XR YR SP'
                  .BY '  NV-BDIZC' $00
;
                  .BA $C214
;
BREAK             CLD
                  LDA #08
                  STA IO.NR
                  LDA #04
                  STA PRINTNR
                  LDA #06
                  STA BORDER
                  STA BKGRND
BREAK1            LDA #03
                  STA COLOR
                  LDX #$05
BREAK2            PLA
                  STA PCHSAVE,X
                  DEX
                  BPL BREAK2
                  LDA PCLSAVE
                  BNE BREAK3
                  DEC PCHSAVE
BREAK3            DEC PCLSAVE
                  TSX
                  STX SPSAVE
                  LDA #'R'
                  JMP CMDSTORE
;
GETSTART          JSR GETRET
                  BEQ GETSTRTS
GETSTART1         JSR GETADR1
                  STA PCLSAVE
                  LDA *PCH
                  STA PCHSAVE
GETSTRTS          RTS
;
GET3ADR           LDX #$A4
                  JSR GETADR
                  JSR GETADR
                  BNE GETADR
;
GET1.2ADR         JSR GETADR1
                  LDA #$FE
                  STA *$FD
                  LDA #$FF
                  STA *$FE
                  JSR GETRET
                  BNE GETADR
                  STA TASTBUF
                  INC *$C6
                  RTS
;
GET2ADR           JSR GETADR1
                  .BY $2C
GETADR1           LDX #$FB
;
GETADR            JSR GETBYT
                  STA *$01,X
                  JSR GETBYT1
                  STA *$00,X
                  INX
                  INX
                  RTS
;
GETBYT            JSR GETCHRERR
                  CMP #$20
                  BEQ GETBYT
                  CMP #$2C
                  BEQ GETBYT
                  BNE ASCHEX
;
GETBYT1           JSR GETCHRERR
ASCHEX            JSR ASCHEX1
                  ASL A
                  ASL A
                  ASL A
                  ASL A
                  STA *$B4
                  JSR GETCHRERR
                  JSR ASCHEX1
                  ORA *$B4
                  RTS
ASCHEX1           CMP #$3A
                  BCC ASCHEX2
                  ADC #$08
ASCHEX2           AND #$0F
                  RTS
;
SKIPSPACE         JSR GETCHRERR
                  CMP #$20
                  BEQ SKIPSPACE
                  DEC *$D3
                  RTS
;
GETRET            JSR CHRIN
                  DEC *$D3
                  CMP #$0D
GETBRTS           RTS
;
GETCHRERR         JSR CHRIN
                  CMP #$0D
                  BNE GETBRTS
ERROR             LDA #'?'
                  JSR CHROUT
EXECUTE           LDX SPSAVE
                  TXS
                  LDX #$00
                  STX *$C6
                  JSR RETURN
                  LDA ($D1,X)
                  CMP #'''
                  BEQ EXEC1
                  CMP #':'
                  BEQ EXEC1
                  CMP #';'
                  BEQ EXEC1
                  CMP #','
                  BEQ EXEC1
                  LDA #'.'
                  JSR CHROUT
EXEC1             JSR GETCHRERR
                  CMP #'.'
                  BEQ EXEC1
CMDSTORE          STA *COMMAND
                  AND #$7F
                  LDX #CMDS-CMDTBL
CMDSEARCH         CMP CMDTBL-1,X
                  BEQ CMDFOUND
                  DEX
                  BNE CMDSEARCH
                  BEQ ERROR
CMDFOUND          JSR CMDEXEC
                  JMP EXECUTE
;
CMDEXEC           TXA
                  ASL A
                  TAX
                  INX
                  LDA CMDS-2,X
                  PHA
                  DEX
                  LDA CMDS-2,X
                  PHA
                  RTS
;
HEXOUT            LDA *PCH
                  JSR HEXOUT1
                  LDA *PCL
HEXOUT1           PHA
                  LSR A
                  LSR A
                  LSR A
                  LSR A
                  JSR HEXOUT2
                  PLA
                  AND #$0F
HEXOUT2           CMP #$0A
                  BCC HEXOUT3
                  ADC #$06
HEXOUT3           ADC #$30
                  JMP CHROUT
;
CHARRET           LDA #$0D
CHARR1            JSR CHROUT
                  TXA
                  JMP CHROUT
;
SPACE2            JSR SPACE
SPACE             LDA #$20
                  JMP CHROUT
;
RETURN            LDA #$0D
                  JMP CHROUT
;
PRINT             STA *$BB
                  STY *$BC
                  LDY #$00
PRINT1            LDA ($BB),Y
                  BEQ PRINT2
                  JSR CHROUT
                  INY
                  BNE PRINT1
PRINT2            RTS
;
PCINC             INC *PCL
                  BNE PCRTS
                  INC *PCH
PCRTS             RTS
;
;**********
EXIT              LDA #$0E
                  STA COLOR
                  STA BORDER
                  LDA #$06
                  STA BKGRND
                  LDA #$37
                  STA *$01
                  LDX SPSAVE
                  TXS
                  JMP READY
;
;**********
REGISTER          LDY #H,REGHEAD
                  LDA #L,REGHEAD
                  JSR PRINT
                  LDX #';'
                  JSR CHARRET
REGISTER1         LDA PCHSAVE
                  STA *PCH
                  LDA PCLSAVE
                  STA *PCL
                  JSR HEXOUT
                  JSR SPACE
                  LDX #$FB
REGISTER2         LDA SRSAVE-$FB,X
                  JSR HEXOUT1
                  JSR SPACE
                  INX
                  BNE REGISTER2
                  LDA SRSAVE
                  JMP CHANGBIN
;
;**********
SEMIS             JSR GETSTART1
                  LDX #$FB
SEMIS1            JSR GETCHRERR
                  JSR GETBYT1
                  STA SRSAVE-$FB,X
                  INX
                  BNE SEMIS1
                  JSR SPACE
                  LDA SRSAVE,X
                  JMP CHANGBIN
;
CHANGBIN          STA *FLAG
                  LDA #$20
                  LDY #$09
CHANGB1           JSR CHROUT
                  ASL *FLAG
                  LDA #$30
                  ADC #$00
                  DEY
                  BNE CHANGB1
                  RTS
;
;**********
GO                JSR GETSTART
GO1               LDX SPSAVE
                  TXS
                  LDX #$FA
GO2               LDA PCHSAVE-$FA,X
                  PHA
                  INX
                  BNE GO2
                  PLA
                  TAY
                  PLA
                  TAX
                  PLA
                  RTI
;**********
MEMDUMP           JSR GET1.2ADR
MEMDUMP1          LDX #':'
                  JSR CHARRET
                  JSR HEXOUT
                  LDY #32
                  LDX #$00
MEMDUMP2          JSR SPACE
                  LDA (PCL,X)
                  JSR HEXOUT1
                  LDA (PCL,X)
                  JSR ASCII
                  BNE MEMDUMP2
                  JSR CONTIN
                  BCC MEMDUMP1
                  RTS
;**********
COLON             JSR GETADR1
                  LDY #32
                  LDX #$00
COLON1            JSR GETCHRERR
                  JSR GETBYT1
                  STA (PCL,X)
                  CMP (PCL,X)
                  BEQ COLON2
                  JMP ERROR
COLON2            JSR ASCII
                  BNE COLON1
                  RTS
;
ASCII             CMP #$20
                  BCC ASCII1
                  CMP #$60
                  BCC ASCII2
                  CMP #$C0
                  BCC ASCII1
                  CMP #$DB
                  BCC ASCII3
ASCII1            LDA #'.'
ASCII2            AND #$3F
ASCII3            AND #$7F
                  STA ($D1),Y
                  LDA COLOR
                  STA ($F3),Y
                  JSR PCINC
                  INY
                  CPY #40
                  RTS
;
CONTIN            JSR TASTE
                  JMP CMPEND1
;
CMPEND            JSR PCINC
CMPEND1           LDA *PCL
                  CMP *$FD
                  LDA *PCH
                  SBC *$FE
                  RTS
;
TASTE             JSR PRINTER1
TASTE1            JSR SCANKEY
                  BEQ TASTRTS
TASTE2            JSR SCANKEY
                  BEQ TASTE2
                  CMP #$20
                  BNE TASTRTS
                  STA TASTBUF
                  INC *$C6
TASTRTS           RTS
;
SCANKEY           JSR GETIN
                  PHA
                  JSR STOPT
                  BEQ STOP
                  PLA
SCANRTS           RTS
STOP              JMP EXECUTE
;
PRINTER1          LDY #40
PRINTER           BIT *COMMAND
                  BPL SCANRTS
                  STY *$C8
                  STY *$D0
                  LDA #$FF
                  JSR CLOSE
                  LDA #$FF
                  STA *$B8
                  STA *$B9
                  LDA PRINTNR
                  STA *$BA
                  JSR OPEN
                  LDX #$00
                  STX *$D3
                  DEX
                  JSR CHKOUT
PRLOOP            JSR CHRIN
                  JSR CHROUT
                  CMP #$0D
                  BNE PRLOOP
                  JSR CLRCHN
                  LDA #$91
                  JMP CHROUT
;
;
;
;
;
;
;***************************
;*                         *
;*        S M O N          *
;*                         *
;* MASCHINENSPRACH-MONITOR *
;*                         *
```

Get Koala Pic

Wenn Sie sowohl einen C 64 als auch das Koala Pad besitzen, dann können Sie jetzt Koala-Bilder in ganz »normale« Basic-Programme einbeziehen.

Sie möchten Koala-Bilder auch ohne Koala laden und zeigen können. Wahrscheinlich haben Sie das Programm am Ende der Anleitung abgetippt, wie ich. Aber dann dachten Sie vielleicht auch, wie schön es wäre, wenn es möglich wäre, die Bilder der Reihe nach, wie eine Dia-Show abzurufen. Dann müßte man nicht mehr STOP-RETURN drücken, und unser Basic-Programm könnte weitergehen. Somit hätte man vielfältige Möglichkeit, Koala-Bilder anzuwenden! Zum Beispiel könnte man:

1....mit Koala ein Titelbild machen

2....mit Koala ein Abenteuer-Programm schreiben; denn: Wenn der Anwender einen Text lesen muß, während das Bild geladen wird, ergibt sich praktisch keine »Wartezeit« für ihn!

3....mit Koala ein Lernprogramm schreiben.

Tja, so ähnlich waren meine Gedankengänge. Aber meine Programmierkenntnisse in Basic nützen nichts; so ein Programm muß in Maschinensprache geschrieben werden! Also telefonierte ich mit einigen C 64-Besitzer-Kollegen herum und wurde fündig: Michael M. Meiszl war schon dabei, an so einem Programm zu basteln. Er paßte es meinen Wünschen an und stellte es in Rekordzeit fertig. Und nun können Sie alles oben Erwähnte tatsächlich tun, und noch viel mehr.

Listing 1 ist das Hauptprogramm. Es wird als erstes geladen. Listing 2 ist ein kleines Anwenderbeispiel, damit Sie sehen, wie die SYS-Befehle in ein normales Basic-Programm eingebaut werden. Bild 1 ist eine Übersicht, welcher SYS-Befehl nun was bewirkt. Falls Sie zwei Laufwerke besitzen, dann können Sie wahlweise 8 oder 9 eingeben.

SYS-BEFEHLE:KOALA PER BASIC	
SYS 52500 +BILD-NAME+,8	LOAD+SHOW Ende:Leertaste
SYS 52503	NUR ANZEIGE
SYS 52506	BILDSCH.NRML
SYS 52509 +BILD-NAME+,8	BILD wird nur geladen
IDEE+ARTIKEL:BIRKENBIHL PROGRAMM VON:M.M.MEISZL	

Bild 1. Die SYS-Befehle im Überblick

Das Hauptprogramm ist sehr anwenderfreundlich: Es geht kein Basic-Speicherplatz verloren. Da jedes Bild einzeln geladen und danach alle Zeiger auf Null gesetzt werden, kann man, der Reihe nach, praktisch unzählige Bilder hintereinander im selben Basic-Programm zeigen. Hierzu muß der Anwender nur die Bilder-Diskette einlegen. Sollten Sie mehrere Bild-Disketten benützen wollen, so genügt ein PRINT-Hinweis auf dem Bildschirm, welche Diskette jetzt eingelegt werden muß. Da die Bilder per Leertaste »beendet« werden, kann das Programm nach einem Bild ganz normal weiterlaufen. Ein SYS-Befehl ermöglicht das Nochmalzeigen des letzten Bildes. Dies kann bei Lernprogrammen von Vorteil sein. Ein anderer SYS-Befehl ermöglicht das Laden ohne Zeigen, so daß dieses Bild dann blitzschnell abgerufen werden kann, da es »unsichtbar« bereits »besteht«. Wenn Sie die Tabelle griffbereit neben sich legen, haben Sie die SYS-Befehle bald im Griff.

Listing 3. Eine Dia-Show auf dem C 64

```
10 REM DEMO #2 ZEIGE BILDER HINTEREINAND
ER WIE EINE DIAS-SHOW
12 REM C MICHAEL M. MEISZL
14 :
20 READA$:IFA$="*"THENRESTORE:GOTO20
30 SYS52509,A$:REM LADE BILD
40 SYS52503::::REM ZEIGE BILD
50 GOTO20
80 REM ==================================
===================
82 REM ACHTUNG:
84 REM IN DIE DATAZEILEN TRAGEN SIE SPAE
TER IHRE EIGENEN BILDNAMEN EIN!
85 REM VORLAEUFIG KOENNEN SIE DIES PROGR
AMM MIT DER OROGINAL-drETTE dXEN
86 REM WICHTIG: DAS "*" MUSS AM ENDE DER
 TITEL STEHEN!
96 REM ==================================
===================
100 DATAABC'S,VAN,VANPROPS,FARM,FARMANIM
,JUNGLE,MONKEY,CASTLE,DRAGON,TREE,*
READY.
```

Noch ein paar Koala-Tips

1. Im Menü finden Sie »INIT DISK«. Damit ist nicht »initialisieren« sondern formatieren gemeint. Trotzdem: Bei Disketten, die mit Koala formatiert wurden, ging jeder zweite Versuch, ein Bild zu laden schief: Dabei stieg dann jedesmal der Computer aus, so daß nicht nur das Bild verloren war, sondern man Koala neu laden mußte, und so weiter.

2. Ich habe bereits des öfteren gelesen, daß die Handhabung im ZOOM-Modus so schwierig sein soll. Mein Vorschlag: Setzen Sie den Pfeil an die richtige Stelle und drücken Sie dann kurz auf die Taste oben. Auf diese Weise kann man punktweise (ebenfalls sehr schnell in lauter Einzelpunkt-Aktionen) und extrem akkurat arbeiten!

3. Wenn Sie ein größeres Stück sauber »radieren« wollen (was durch DRAW mit Hintergrundfarbe leicht ist), dann hilft Ihnen ZOOM ebenfalls: »Radieren« Sie mit ZOOM erst die Ränder mit dem feinsten »Pinsel«, dann können Sie mit einem breiten »Pinselstrich« den Mittelteil schnell weg-malen!

4. Wenn Sie etwas Riskantes ausprobieren wollen, was möglicherweise Ihr Bild kaputtmachen müßte (erst »radieren«, dann das Neue zeichnen, dann erst sehen, ob es Ihnen gefällt), dann rate ich Ihnen: Kopieren Sie das gesamte Bild per COPY (dem ein SWAP vorausgehen muß) auf den zweiten Bildschirm. Jetzt können Sie nach Herzenslust experimentieren, denn Ihr Bild existiert ja noch »im Original« auf der anderen Grafikseite.

So, jetzt hoffe ich nur noch, daß Sie in Zukunft noch mehr Spaß mit Koala haben werden.

(Vera F. Birkenbihl/aa)

Listing 1. Das Hauptprogramm, um Koala-Bilder in eigene Basic-Programme einzubinden.

```
100 REM DISPLAY KOALA PIC V2.1
110 REM (C) 1984 BY MICHAEL A. MEISZL
120 REM ============================
122 : REM ACHTUNG: DAS PROGRAMM ENTHAELT
 PRUEFSUMMEN, SO DASS SIE FEHLER ...
124 REM ... LEICHTER dMEN KOENNEN, BES.
 IN DEN DATA ZEILEN!
126 REM ================================
=======================================
130 A=52500:E=A+335:ZL=9990:POKE53281,0:
POKE53280,0
140 PRINT"▒▒▒KOALA PAINTER DISPLAY▒▒▒
150 PRINT"▒SYS 52500,"CHR$(34)"NAME"CHR$
(34)"[,GERAETENUMMER]
160 PRINT,"▒=> LOAD + ANZEIGE":PRINT,"(E
NDE MIT 'SPACE')
170 PRINT"▒▒SYS 52503
180 PRINT,"▒=> NUR ANZEIGE
190 PRINT"▒▒SYS 52506
200 PRINT,"▒=> BILDSCHIRM WIEDER NORMAL
210 PRINT"▒▒SYS 52509,"CHR$(34)"NAME"CHR
$(34)"[,GERAETENUMMER]
220 PRINT,"▒=> BILD LADEN":PRINT:PRINT
230 FORI=ATOESTEP8:ZL=ZL+10
240 CH=0:FORJ=0TO7:READX:CH=CH+X:POKEI+J
,X:NEXT
250 READCK:IFCK<>CHTHENPRINT"▒DATA ERROR
 IN ZEILE"ZL:END
260 NEXT:PRINT"▒PROGRAMM BEREIT
10000 DATA76,32,205,76,145,205,76,79, 89
4
10010 DATA206,76,47,205,32,47,205,32, 85
0
10020 DATA145,205,32,228,255,201,32,208,
 1306
10030 DATA249,240,235,32,66,206,32,87, 1
147
10040 DATA226,201,9,144,3,76,88,182, 929
10050 DATA160,9,169,32,153,97,206,136, 9
62
10060 DATA208,250,177,187,153,98,206,200
, 1479
10070 DATA196,183,144,246,169,15,162,91,
 1206
10080 DATA160,206,32,189,255,162,8,32, 1
044
10090 DATA121,0,201,44,208,3,32,155, 764
10100 DATA183,169,111,160,0,32,186,255,
1096
10110 DATA32,131,205,169,0,162,0,160, 85
9
10120 DATA160,32,213,255,8,32,138,205, 1
043
10130 DATA40,176,1,96,76,156,225,165, 93
5
10140 DATA1,41,254,133,1,96,165,1, 692
10150 DATA9,1,133,1,96,173,17,208, 638
10160 DATA41,239,141,17,208,32,131,205,
1014
10170 DATA162,191,160,64,32,56,206,162,
1033
10180 DATA200,160,0,32,61,206,162,195, 1
016
10190 DATA160,39,32,14,206,162,195,160,
968
10200 DATA40,32,56,206,162,216,160,0, 87
2
10210 DATA32,61,206,162,199,160,15,32, 8
67
10220 DATA14,206,173,16,199,141,33,208,
990
10230 DATA141,32,208,173,17,208,9,34, 82
2
10240 DATA141,17,208,173,22,208,41,223,
1033
10250 DATA9,16,141,22,208,169,40,141, 74
6
10260 DATA24,208,162,160,160,0,32,56, 80
2
10270 DATA206,162,224,160,0,32,61,206, 1
051
10280 DATA162,191,160,63,32,14,206,173,
1001
10290 DATA0,221,41,248,141,0,221,173, 10
45
10300 DATA17,208,9,16,141,17,208,76, 692
10310 DATA138,205,142,106,206,140,107,20
6, 1250
10320 DATA160,0,177,251,145,253,165,252,
 1403
10330 DATA205,106,206,208,8,165,251,205,
 1354
10340 DATA107,206,208,1,96,230,251,208,
1307
10350 DATA2,230,252,230,253,208,227,230,
 1632
10360 DATA254,76,22,206,134,252,132,251,
 1327
10370 DATA96,134,254,132,253,96,32,121,
1118
10380 DATA0,201,44,208,3,76,115,0, 647
10390 DATA76,8,175,32,129,255,173,0, 848
10400 DATA221,9,3,141,0,221,96,129, 820
10410 DATA80,73,67,32,63,32,0,0, 347
READY.
```


Listing 2. Ein Demo Programm

```
10 REM DEMO 1 KOALA BILDER PER BASIC-PRO
GRAMM ZEIGEN
12 REM C MICHAEL M. MEISZL,1984
14 :
20 PRINT"▒KOALA PAINTER DISPLAY DEMO 1
30 A$="":INPUT"▒▒▒▒BILDNAME";A$
40 IFA$=""THEN END
45 IFA$=B$THEN100
50 SYS(52500),A$
60 B$=A$:GOTO20
100 REM DEMO ANZEIGE EINES GESPEICHERTEN
 BILDES
110 SYS52503:REM BILD NOCHMAL
120 FORI=1TO2000:NEXT
130 SYS52506:REM BILD ABSCHALTEN
140 GOTO20
READY.
```




Der VC 20 als Laterna Magica

Laterna Magica macht Spiele beweglicher und professioneller. Man kann Zeichen und zusammengesetzte Bilder nicht nur entwickeln, sondern auch bewegen, pulsieren, rotieren oder explodieren lassen und in eigene Programme einbauen.

»Laterna Magica« ist für jeden VC 20 gedacht, sei er erweitert oder nicht. Als Peripherie wird ein Floppy-Disk-Laufwerk VC-1541 vorausgesetzt. Das Programm kann aber für die Datasette umgeschrieben werden, indem die Gerätenummer 8 in Zeile 60 des DATA-Laders in eine 1 umgewandelt wird. Außerdem muß es dann im Hauptprogramm heißen:

490 PRINT"SAVE"CHR$(34)"   "CHR$(34)",1,1"

Das Programm (Listing 1) ist nicht mehr zu erweitern, es können also auch keine REM-Zeilen mehr eingefügt werden. Es sieht zwar als Listing nicht sehr umfangreich aus, jedoch mußte das Basic-Ende heruntergesetzt werden, um das Maschinenprogramm, den neuen Zeichengenerator und den Speicherplatz, in dem wir später unsere Ersatzbilder speichern, vor Überschreibung zu schützen. Die Erzeugung von Grafik beim VC 20 beruht ja darauf, die Zeichen auf dem Bildschirm, die aus 88 Punkten aufgebaut sind, in ihrem Aufbau zu ändern. Bei einem Zeichen hieße es, 8 Byte gegen 8 andere auszutauschen. Bei 9 Zeichen (das ist die Zeichenanzahl, mit der »Lanterna Magica« arbeitet) werden es 72 Byte. Versuchte man das Problem mit Basic zu lösen, müßte man bei jedem Tauschvorgang eine Zählschleife von 1 bis 72 mit einigen POKEs dazwischen in Kauf nehmen. So etwas kann Action-Spiele ganz schön langsam machen. Ich habe diese Routine daher in Maschinensprache geschrieben. So genügt ein SYS-Aufruf, und 9 Zeichen auf dem Bildschirm ändern sofort ihre Form.

Ausgeführt werden kann der Tausch nur, wenn der Zeichengenerator nicht wie üblich im ROM liegt. Das ist also die erste Aufgabe des Maschinenprogramms. Es legt den Zeichengenerator ins RAM ab $1C00(entspricht dezimal 7168). Danach holt es sich aus dem Reserve-Bildschirmspeicher ($1A00 bis $1BFF, dezimal 6656 bis 7167) die ersten 72 Byte und legt sie im Bildschirmspeicher ab. Das ist zunächst alles. Beim nächsten Aufruf verfährt es mit den folgenden 72 Byte in gleicher Weise. Das geht siebenmal so, danach werden wieder die ersten 72 Byte geholt und so weiter.

Wichtig ist, daß vor jedem Aufruf in die Speicher $FB und $FC (dezimal 251 und 252) ein Wert von maximal 7 gePOKEt wird. An diesem Wert erkennt das Maschinenprogramm, wievielmal es die Bilder austauschen soll, bis es wieder beim ersten beginnt. Das bedeutet, daß man auch »Filme« mit nur drei Bildern laufen lassen kann, indem man den Wert 3 in diese Speicher POKEt.

Der Bildeditor

Um diese besprochenen Bilder zu entwickeln, wären langwierige Berechnungen und viel Tastaturarbeit zum Eingeben der bis zu 504 Werte notwendig. Deswegen habe ich ein Programm geschrieben, das diese Arbeit übernimmt. (Listing 2).

Der Umgang mit dem Bildeditor ist recht einfach. Nach RUN erscheint in der oberen Hälfte des Bildschirmes eine Maske, in der die Zeichen aufgebaut werden und in der unteren Hälfte das Menü (siehe Bild).

Das Feld rechts oben zeigt das erste Bild. Man sieht zwar zunächst nur ein abstraktes Gebilde, das liegt aber nur daran, daß noch keine Zeichen entwickelt wurden. Links daneben stehen die gleichen Zeichen in Klarschrift. In beiden Feldern sind die gleichen Zeichen in der linken oberen Ecke schwarz, um herauszustellen, daß dieses Zeichen bearbeitet werden kann.

Mit F1 kann man nun ein Zeichen auswählen, das verändert werden soll. Dabei wandert das schwarze Feld jeweils eine Ziffer weiter.

Weiterhin kann man über den beiden Feldern die Bildnummer ablesen. Mit F3 können wir zwischen sieben Bildern wählen. Die Bilder werden jeweils im rechten Feld gezeigt. Es entsteht so schon ein Zeichentrickeffekt.

Hat man nun seine Wahl getroffen, kann man mit F7 in den Entwicklungsmodus wechseln. Der Cursor steht nun in der linken oberen Ecke des großen Entwicklungsfeldes. Er läßt sich

```
10 REM * LATERNA-MAGICA-DATA-LOADER *
12 REM
15 REM ==============================
16 REM
20 POKE52,25:POKE56,25:CLR
30 FORI=6400TO6640:READA:POKEI,A:S=S+A:N
EXT
40 IFS<>26746THENPRINT"DATA-FEHLER!":END
50 PRINT"OK!"
60 A$="LOAD"+CHR$(34)+"LATERNA MAGICA"+C
HR$(34)+",8"
70 PRINTCHR$(147);A$:PRINTCHR$(19):PRINT
:PRINT:PRINT:PRINT:PRINT:PRINT"RUN"
75 POKE631,19:POKE632,13:POKE633,13:POKE
198,3:NEW
80 DATA173,5,144,41,240,9,15,141,5,144,1
98,251,162,72,189,71
90 DATA26,157,127,29,202,208,247,173,15,
25,24,105,72,141,15,25
100 DATA173,16,25,105,0,141,16,25,165,25
1,208,14,169,255,141,15
110 DATA25,169,25,141,16,25,165,252,133,
251,96,169,0,133,204,32
120 DATA245,241,240,251,76,211,25,234,72
,120,165,207,240,12,165,206
130 DATA174,135,2,160,0,132,207,32,161,2
34,88,104,32,122,242,166
140 DATA214,224,8,48,2,162,0,164,211,192
,8,48,2,160,0,32
150 DATA12,229,76,175,25,32,129,229,162,
0,142,60,3,160,0,177
160 DATA209,201,160,240,9,169,1,24,109,6
0,3,141,60,3,192,7
170 DATA240,3,14,60,3,200,192,8,48,229,3
2,19,229,232,173,60
180 DATA3,32,199,25,224,8,240,6,32,12,22
9,76,120,25,96,32
190 DATA19,229,152,72,138,72,169,1,133,2
04,32,117,25,104,170,104
200 DATA168,32,12,229,76,59,25,157,7,26,
157,135,29,96,104,168
210 DATA104,170,96,201,13,240,3,76,72,25
,120,165,207,240,12,165
220 DATA206,174,135,2,160,0,132,207,32,1
61,234,88,169,1,133,204
230 DATA96

READY.
```

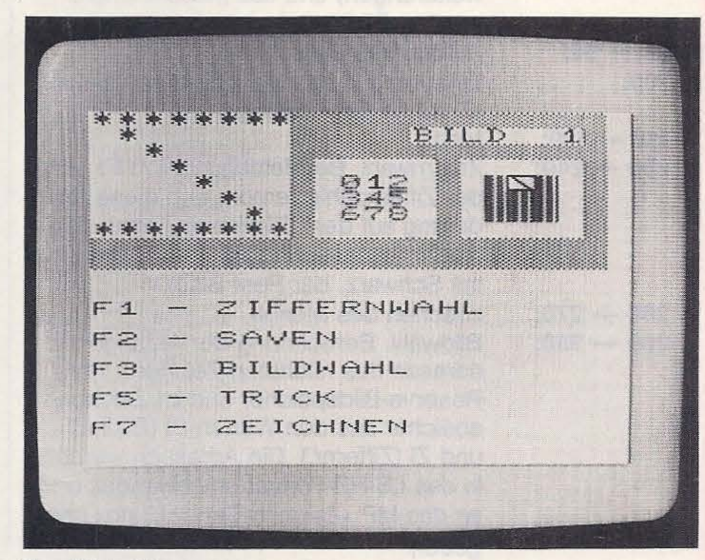
Listing 1. »Laterna Magica« Basic-Lader

```
10 POKE36869,PEEK(36869)AND240OR15
20 IF(PEEK(36866)OR127)=255THENAZ=38400
30 IF(PEEK(36866)OR127)<>255THENAZ=37888
40 BI=0:POKE251,7:POKE252,7:POKE6415,255
:POKE6416,25:SYS6400
50 PRINT"▒";:FORI=1TO11:PRINT"▒░░░░░░░░░░░░░░░
":NEXT
60 PRINT"▒";:FORI=1TO8:PRINT"▒             ":
NEXT
70 PRINT"▒▒▒▒"TAB(9)"▒        ░         ":PRINT
TAB(9)"▒ 012 ░ █012▒ █"
80 PRINT TAB(9)"▒ 345 ░ █345▒ █"
90 PRINT TAB(9)"▒ 678 ░ █678▒ █":PRINTTA
B(9)"▒     ░     ":GOTO200
100 PRINT"▒▒";:SYS6459:POKE204,1
110 PRINT"▒":WV=11:GOSUB510
120 PRINT"▒F1 - ZIFFERNWAHL"
130 PRINT"▒▒F2 - SAVEN"
140 PRINT"▒▒F3 - BILDWAHL"
150 PRINT"▒▒F5 - TRICK"
160 PRINT"▒▒F7 - ZEICHNEN"
170 GETA$:IFA$=""THEN170
180 IFASC(A$)<133ORASC(A$)>137THEN170
190 ONASC(A$)-132GOTO200,280,360,250,450
200 ZI=ZI+1:IFZI>9THENZI=1
210 FORJ=1TO3:FORI=0TO2:HH=AZ+104+I+K:HI
=HH-6:TE=TE+1:IFTE=ZITHEN230
220 POKEHH,5:POKEHI,5:GOTO240
230 POKEHH,0:POKEHI,0
240 NEXTI:K=K+22:NEXTJ:K=0:TE=0:PR=1:GOT
O290
250 PRINTCHR$(19);:WV=10:GOSUB510
260 FORI=1TO11:PRINT"▒
N  ":NEXT:PRINT"▒";:IFLE=1THENLE=0:RETUR
270 GOTO100
280 BI=BI+1:IFBI>6THENBI=0
290 PRINT"▒▒▒"TAB(13)"BILD "BI+1
300 X=72*BI+8*(ZI-1)+6655:Q=INT(X/256):R
=X-256*Q
310 POKE6600,R:POKE6601,Q
320 HI=7551+8*(ZI-1):H=INT(HI/256):L=HI-
256*H
330 POKE6603,L:POKE6604,H
340 IFPR=1THENPR=0:GOTO110
350 SYS6400:GOTO110
360 LE=1:GOSUB250:PRINT"▒";:WV=11:GOSUB5
10:PRINT"▒TEMPO        0=MAX."
370 PRINT:PRINTTAB(7)"▒1000=KLEIN"
380 PRINT:PRINT"▒█100██████";:INPUT"▒";T
E
390 PRINT"▒":WV=8:GOSUB510:PRINTTAB(9)"0
12"
400 PRINTTAB(9)"345"
410 PRINTTAB(9)"678"
420 PRINT"▒▒▒▒▒>> STOP << MIT SPACE!"
430 SYS6400:FORI=0TOTE:NEXT:IFPEEK(197)=
32THENLE=1:GOSUB250:CLR:GOTO10
440 GOTO430
450 POKE680,PEEK(43):POKE681,PEEK(44):PO
KE682,PEEK(45):POKE683,PEEK(46)
460 POKE36869,PEEK(36869)AND240
470 PRINT"▒BITTE NAMEN EINSETZEN:"
480 PRINT"P┌43,0:P┌44,26:P┌45,0:P┌46,28:
P┌198,1:P┌631,13:";
490 PRINT"SAVE"CHR$(34)"@0:
"CHR$(34)",8,1"
500 PRINT"▒▒▒P┌43,P‾(680):P┌44,P‾(681):P
┌45,P‾(682):P┌46,P‾(683):RUN":END
510 FORI=1TOWV:PRINTCHR$(17);:NEXT:RETUR
N
READY.
```

Listing 2. »Laterna Magica« Grafik-Editor

voll steuern, jedoch nicht über den Feldrand hinaus. Tippt man nun, mehr oder weniger willkürlich, Buchstaben oder Zeichen in das Feld, sieht man gleichzeitig an der vorgewählten Ziffer im rechten Feld das Ergebnis. Man sollte nur nicht den Fehler machen, CLR/HOME oder INST/DEL zu drücken. Das kann den Bildschirmaufbau durcheinander bringen.

Mit RETURN kommt man ins Menü zurück.



Der Bildeditor

Ist eine Bild- oder Zeichenfolge fertiggestellt, kann man den Trickfilm mit F5 ausprobieren. Im Menüfeld wird die Geschwindigkeit des Ablaufes erfragt. Eine gute Geschwindigkeit von 100 ist vorgegeben. Ein Wert von Null ergibt die größte Geschwindigkeit. In der anderen Richtung wird man selbst merken, wenn es langweilig wird. Mit RETURN wird gestartet, und mit der SPACE-Taste kann der Vorgang abgebrochen werden.

Es bleibt noch die F2-Funktion zu erklären. Bei Druck auf F2 erscheinen ein paar im ersten Augenblick wirr anmutende Befehle und die Bitte, den Namen der Bildfolge einzutragen. Das bedeutet, daß in der ersten Zeile (mit den vielen POKEs und dem SAVE) in die Lücke ein Name eingetragen werden soll. Es ist auch angebracht, das zweite Anführungszeichen an den Namen heranzuziehen. Dann drückt man RETURN, und alles geht wieder automatisch. Die Bildfolge wird als Maschinenprogramm geSAVEt und das Programm startet von neuem.

Zur Eingabe des Programmes: Bei Verwendung der Datasette muß zunächst der DATA-Lader (Listing 1) abgetippt und abgespeichert werden und danach Listing 2, weil der DATA-Lader selbständig das nächste Programm nachlädt. Verwendet man ein Floppy-Laufwerk, ist die Reihenfolge unerheblich.

Will man mit dem Programm arbeiten, muß erst der DATA-Lader geladen werden. »RUN« startet den Lader, der das Hauptprogramm lädt.

Bildfolgen in eigenen Programmen

Will man die Trickfolge im selbstgemachten Programm anwenden, muß natürlich das Machinenprogramm im Speicher vorhanden sein. Dazu lädt man den DATA-Lader, löscht die Zeilen 60-75, setzt in Zeile 60 ein NEW und startet den Lader. Damit ist das Maschinenprogramm im Bereich dezimal 6400 bis 6640 abgelegt und vor Überschreibung geschützt. Der Lader löscht sich selbst. Außerdem ist auch der Bereich geschützt, in den nun die selbstgemachte Bildfolge mit »LOAD "Name",8,1« oder »LOAD "Name",1,1« geladen werden kann.

Damit keine Fehlermeldungen auftreten, gibt man jetzt noch den Befehl NEW ein und drückt die Return-Taste. Jetzt kann man das eigene Programm laden oder eintippen.

Programm-Übersicht

10 — 40:	Hier wird die Anfangsadresse des Zeichenspeichers nach Adresse 7168 verlegt, der Beginn des Farb-RAM ermittelt (unterschiedliche Werte bei GV und Erweiterungen) und das Maschinenprogramm »Bildtausch« initialisiert.
50 — 90:	Aufbau des Bildschirms
100:	Einsprung in das Maschinenprogramm »Zeichen-Entwicklung«
110 — 190:	Menü
200 — 240:	Ziffernwahl. Bei Betätigung von F1 wird der Ziffernzähler erhöht und diese Änderung auf dem Bildschirm kenntlich gemacht (Unterlegung der gewählten Ziffer mit Schwarz, der Rest Blau).
260 — 270:	Löschen des Menüs
280 — 350:	Bildwahl. Berechnung der Anfangsadresse des nächsten Zeichens im Reserve-Bildspeicher und im Zeichenspeicher aus den Werten BI (Bildnr.) und ZI (Ziffernr.). Die Adressen werden in das LB-HB-Format umgewandelt und an das MP »Zeichen-Entwicklung« übergeben.
360 — 440:	Trick. Der Bildschirm wird neu aufgebaut. Die Hauptsache ist jedoch die Zeile 430. Dort wird immer wieder das Maschinenprogramm »Bildtausch« aufgerufen, das einen Mini-Trickfilm auf dem Bildschirm erzeugt.
450 — 500:	Basic-Anfang und -Ende werden in den Bytes 680-683 abgelegt. Der Zeichenspeicher wird normalisiert. Es wird eine Maske zum Abspeichern des Reserve-Bildspeichers ausgegeben. Maske: Basic-Anfang und -Ende werden gleich Reserve-Bildspeicher-Anfang und -Ende gesetzt. In den Tastaturpuffer wird ein RETURN = CHR$(13) gePOKEt, damit nach dem Abspeichern die zweite Zeile der Maske abgearbeitet wird, die Basic-Anfang und -Ende wieder in den Ausgangszustand bringt. Mit dem »RUN« am Ende der Zeile startet das Programm wieder.
510:	Unterprogramm zur Ausgabe von mehreren »Cursor Down«

Die wichtigsten Variablen

AZ	Beginn des Farb-RAM
BI	Bildzähler
I,J,K	Schleifenzähler
WV	Parameterübergabe für Unterprogramm in Zeile 510
A$	Abfrageergebnis aus dem Menü
ZI	Ziffernzähler
HH,HI	Farbspeicher, die auf dem Bildschirm schwarz erscheinen sollen
TE	Zähler zum Vergleich mit aktueller Ziffer (200 bis 240) und Geschwindigkeit für »Trick« (360 bis 440)
PR	Verzweigungsargument für Unterprogramm (280 bis 350)
LE	Verzweigungsargument für Zeile 260
X	1. Speicherplatz für das nächste zu erstellende Zeichen (im Reserve-Bildspeicher)
Q	HB von X
R	LB von X
HI	Äquivalent zu X (für Zeichenspeicher)
H	HB von HI
L	LB von HI

Das Maschinenprogramm wird übrigens mit SYS6400 aufgerufen.

Und noch einmal: Am Anfang des Programmes, vor dem ersten SYS6400 muß unbedingt eine Zahl zwischen 1 und 7 in 251 und 252 gePOKEt werden, eben die Anzahl der Bilder des Trickfilms. Alle kleineren Zahlen als 1 und alle größeren als 7 richten zwar keinen Schaden an, jedoch bewirken sie eine Einschränkung der Funktion.

Sollte man dies einmal vergessen haben, ist es angebracht, das Maschinenprogramm neu zu laden.

Zum Schluß noch zwei Tips.

1. Bei erweitertem VC 20 sollte man für eigene Programme nicht das Basic-Ende herunter-, sondern den Basic-Beginn heraufsetzen. Damit hat man mehr Speicherplatz für seine Programme. (Zeile 20) des DATA-Laders: POKE44,32:POKE8192,0:CLR).
2. Da bei der Arbeit mit diesem Programm nicht der Original-Zeichensatz vorhanden ist, kann man ihn, wenn gebraucht, über die RVS-ON-Taste darstellen. Dieses Verfahren habe ich selbst im Listing 1 verwendet. (Bernd Schrödter/ev)

Grafik leicht gemacht

Mit einigen neuen Befehlen macht dieses Programm das Plotten von Funktionen zum Kinderspiel. Das mühsame Arbeiten mit den Commodore-Grafikbefehlen ist vorbei, wenn man die neuen Befehle einsetzt.

Das Programm stellt eine Basic-Erweiterung dar, die im Bereich von $0800-$4000 angesiedelt ist. Dieser Bereich enthält das Maschinenspracheprogramm sowie den Grafikbildschirm. Die neuen Befehle und ihre Bedeutung:

HIRES: Dieser Befehl schaltet die hochauflösende Grafik ein.
SCNCLR: Hiermit wird der Grafikbildschirm gelöscht.
TEXT: Um die Grafik wieder auszuschalten wird dieser Befehl benutzt. Am Programmende schaltet die hochauflösende Grafik automatisch ab.
REGION a: Die Cursor- beziehungsweise Plotfarbe wird festgelegt. Der Parameter a darf Werte von 0 bis 15 annehmen.
COLOR a,b,c: Bestimmt die Hintergrundfarbe (a), die Rahmenfarbe (b) und die Zeichenfarbe (c).
PLOT x,y: Dieser Befehl setzt einzelne Punkte auf dem Grafikbildschirm. Die Parameter x und y geben die Position des Punktes an. Der Ursprung des Koordinatensystems liegt in der linken oberen Bildschirmecke. X darf Werte von 0 bis 319, y Werte von 0 bis 199 annehmen. Bei Überschreiten dieser Bereiche gibt der Computer jedoch keine Fehlermeldung aus, wie es bei anderen Programmen meist der Fall ist. Für die Darstellung von Funktionen ist diese Einrichtung jedoch sehr nützlich.
UNPLOT x,y: Löscht einen gesetzten Punkt. Für die Parameter gelten die gleichen Grenzen wie beim Befehl PLOT.

AXES x,y: Zeichnet Koordinatenachsen auf den Grafikbildschirm. Die Parameter x und y bestimmen in diesem Fall den Ursprung des Koordinatensystems.

ONERROR GOTO n: Dieser Befehl fängt Definitionslücken von Funktionen ab, wenn nicht mit dem FPLOT-Befehl gearbeitet wird (siehe dort). Tritt während des Programmablaufs ein Fehler auf, der durch eine Definitionslücke verursacht wurde, wird zur Zeile n verzweigt. Dort wird das Programm ohne Unterbrechung weitergeführt. Die Fehlerausgabe wird sinnvollerweise jedoch nur bei den folgenden Fehlern unterdrückt:

ILLEGAL QUANTITY, DIVISION BY ZERO, OVERFLOW

Bei anderen Fehlerursachen bricht das Programm wie üblich ab.

FPLOT f(x) STEP n: Dieser Befehl stellt das Kernstück des Programms dar. Er ermöglicht das Plotten einer beliebigen Funktion mit einem Befehl. Er setzt voraus, daß das Koordinatensystem seinen Ursprung in der Mitte des Bildschirms hat. Nach STEP kann noch angegeben werden, in welchen Intervallen Punkte für die Funktionsdarstellung berechnet werden sollen. Soll die Funktion sehr genau ausfallen, muß n klein ge-

```
10 REM **************************
20 REM *  FUNKTIONENPLOT  C 64  *
30 REM *    UWE SEIMET          *
40 REM *    WAERDERWEG 47       *
50 REM *    4170 GELDERN 4      *
60 REM *    TELEFON: 02831/7637 *
70 REM **************************
80 REM
90 REM
100 DATA0,29,8,192,7,158,40,50,49,49,55,
41,32,18,40,67,41,32,49,57,56,52
110 DATA32,66,89,32,85,83,0,0,0,7,9,131,
164,87,9,10,10,84,10,141,10,70,85
120 DATA78,75,84,73,79,78,69,78,80,76,79
,84,32,66,89,32,85,83,32,42,42,42
130 DATA42,32,169,55,133,1,160,0,132,251
,132,253,169,160,133,252,169,224
140 DATA133,254,177,251,145,251,177,253,
145,253,200,208,245,230,252,230,254
150 DATA208,239,162,26,189,42,8,157,125,
228,202,208,247,142,217,236,142,218
160 DATA236,169,5,141,53,229,169,175,141
,46,160,169,10,141,47,160,169,76
170 DATA141,55,169,169,126,141,56,169,16
9,10,141,57,169,169,224,141,94,160
180 DATA169,10,141,95,160,169,53,133,1,1
41,214,253,32,24,229,162,11,189,31
190 DATA8,157,0,3,202,16,247,232,160,64,
24,32,3,228,76,154,227,72,73,82,69
200 DATA211,84,69,88,212,83,67,78,67,76,
210,82,69,71,73,79,206,67,79,76,79
210 DATA210,80,76,79,212,85,78,80,76,79,
212,70,80,76,79,212,65,88,69,211
220 DATA69,82,82,79,210,0,25,11,63,11,17
1,13,157,13,95,13,140,11,137,11,91
230 DATA12,250,12,9,175,138,48,58,173,62
,3,208,7,173,64,3,201,88,208,46,224
240 DATA14,240,8,224,15,240,4,224,20,208
,34,173,64,3,201,88,208,7,174,63
250 DATA3,154,76,217,12,173,60,3,133,20,
173,61,3,133,21,32,163,168,174,63
260 DATA3,154,76,174,167,169,0,141,62,3,
164,2,240,7,138,72,32,63,11,104,170
270 DATA76,139,227,166,122,160,4,132,15,
189,0,2,16,7,201,255,240,62,232,208
280 DATA244,201,32,240,55,133,11,201,34,
240,86,36,15,112,45,201,63,208,4
290 DATA169,153,208,37,201,48,144,4,201,
60,144,29,132,113,160,0,132,11,136
300 DATA134,122,202,200,232,189,0,2,56,2
49,158,160,240,245,201,128,208,48
310 DATA5,11,164,113,232,200,153,251,1,1
85,251,1,240,89,56,233,58,240,4,201
320 DATA73,208,2,133,15,56,233,85,208,15
9,133,8,189,0,2,240,223,197,8,240
330 DATA219,200,153,251,1,232,208,240,16
6,122,230,11,200,185,157,160,16,250
340 DATA185,158,160,208,180,160,255,202,
200,232,189,0,2,56,249,192,8,240
350 DATA245,201,128,208,2,240,173,166,12
2,230,11,200,185,191,8,16,250,185
360 DATA192,8,208,226,189,0,2,16,155,76,
9,166,16,66,201,255,240,62,36,15
370 DATA48,58,170,132,73,201,204,176,10,
160,160,132,35,160,158,132,34,208
380 DATA11,233,76,170,160,8,132,35,160,1
92,132,34,160,0,10,240,16,202,16
390 DATA12,230,34,208,2,230,35,177,34,16
,246,48,241,200,177,34,48,8,32,71
400 DATA171,208,246,76,243,166,76,239,16
6,186,142,63,3,32,115,0,201,204,144
410 DATA25,201,213,176,21,32,105,10,76,1
74,167,233,203,10,168,185,244,8,72
420 DATA185,243,8,72,76,115,0,32,121,0,7
6,231,167,165,97,208,3,76,59,169
430 DATA32,121,0,176,209,76,160,168,169,
0,133,13,32,115,0,176,3,76,243,188
440 DATA32,19,177,144,15,205,64,3,240,3,
76,40,175,169,65,160,3,76,162,174
450 DATA76,154,174,201,213,240,3,76,75,1
69,32,166,179,32,115,0,169,137,32
460 DATA255,174,32,138,173,32,247,183,32
,19,166,176,3,76,227,168,165,20,141
470 DATA60,3,165,21,141,61,3,169,137,141
,62,3,96,32,43,188,240,52,16,3,76
480 DATA72,178,32,199,187,165,97,56,233,
129,8,74,24,105,1,40,144,2,105,127
490 DATA133,97,169,4,133,103,32,202,187,
169,92,160,0,32,15,187,169,87,160
500 DATA0,32,103,184,198,97,198,103,208,
233,96,169,59,141,17,208,169,24,141
510 DATA24,208,160,0,162,4,132,253,134,2
54,173,33,208,41,15,145,253,200,208
520 DATA251,230,254,202,208,246,232,134,
2,96,169,0,133,2,169,27,141,17,208
530 DATA169,21,141,24,208,76,68,229,0,0,
64,1,128,2,192,3,0,5,64,6,128,7,192
540 DATA8,0,10,64,11,128,12,192,13,0,15,
64,16,128,17,192,18,0,20,64,21,128
550 DATA22,192,23,0,25,64,26,128,27,192,
28,0,30,1,2,4,8,16,32,64,128,169
560 DATA128,44,169,0,133,151,32,138,173,
32,247,183,32,253,174,32,138,173
570 DATA32,155,188,165,100,208,154,166,1
01,224,200,176,148,165,21,201,1,144
580 DATA8,208,140,165,20,201,64,176,134,
138,74,74,74,10,168,185,80,11,133
590 DATA247,185,81,11,133,248,138,41,7,2
4,101,247,133,247,165,20,41,248,133
600 DATA249,24,169,0,101,247,133,253,169
,32,101,248,133,254,24,165,253,101
610 DATA249,133,253,165,254,101,21,133,2
54,56,233,32,133,252,165,253,133
620 DATA251,70,252,102,251,70,252,102,25
1,70,252,102,251,24,165,252,105,4
630 DATA133,252,160,0,173,134,2,10,10,10
,10,133,97,177,251,41,15,5,97,145
640 DATA251,165,20,41,7,73,7,170,189,130
,11,160,0,36,151,16,5,73,255,49,253
```

Listing »Funktionenplot«

```
650 DATA44,17,253,145,253,96,131,160,0,0
,0,131,32,0,0,0,123,117,194,143,92
660 DATA134,0,0,0,0,133,160,0,0,0,136,32
,0,0,0,135,72,0,0,0,32,166,179,169
670 DATA0,133,151,169,88,141,64,3,166,12
2,164,123,208,1,136,202,152,72,138
680 DATA72,32,138,173,169,67,160,12,32,1
62,187,162,169,32,11,169,201,169
690 DATA208,9,32,251,168,32,115,0,32,138
,173,162,70,160,3,32,212,187,186
700 DATA142,63,3,169,57,160,12,32,162,18
7,120,104,170,104,134,122,133,123
710 DATA72,138,72,162,65,160,3,32,215,18
7,169,72,160,12,32,40,186,169,82
720 DATA160,12,32,103,184,32,247,183,32,
166,173,169,77,160,12,32,40,186,169
730 DATA87,160,12,32,103,184,32,157,11,1
69,65,160,3,32,162,187,169,70,160
740 DATA3,32,103,184,169,62,160,12,32,91
,188,176,179,88,104,104,169,0,141
750 DATA64,3,76,248,168,32,138,173,32,24
7,183,32,253,174,32,138,173,32,155
760 DATA188,165,100,240,3,76,62,11,165,1
01,201,200,176,247,165,21,133,39
770 DATA201,1,144,8,208,237,165,20,201,6
4,176,231,165,20,133,38,169,63,133
780 DATA20,169,1,133,21,166,101,32,184,1
1,198,20,165,20,201,255,208,243,198
790 DATA21,16,239,165,38,133,20,165,39,1
33,21,162,199,134,101,169,0,133,100
800 DATA32,184,11,198,101,166,101,224,25
5,208,245,96,32,158,183,224,16,176
810 DATA66,142,33,208,32,155,183,224,16,
176,56,142,32,208,32,115,0,32,158
820 DATA13,165,2,240,31,173,33,208,41,15
,133,97,160,0,162,4,132,253,134,254
830 DATA177,253,41,240,5,97,145,253,200,
208,245,230,254,202,208,240,96,32
840 DATA158,183,224,16,176,4,142,134,2,9
6,76,72,178,160,0,162,32,132,100
850 DATA134,101,152,145,100,200,208,251,
230,101,202,208,246,96
900 PRINTCHR$(147)
1000 FORI=0TO415:READQ:S1=S1+Q:POKE2048+
I,Q:NEXT
1010 IFS1<>45402THENPRINT"FEHLER IN DEN
DATAS DER ZEILEN 100-300!":END
1100 FORI=416TO801:READQ:S2=S2+Q:POKE204
8+I,Q:NEXT
1110 IFS2<>46267THENPRINT"FEHLER IN DEN
DATAS DER ZEILEN 310-500":END
1200 FORI=802TO1193:READQ:S3=S3+Q:POKE20
48+I,Q:NEXT
1210 IFS3<>42667THENPRINT"FEHLER IN DEN
DATAS DER ZEILEN 510-700!":END
1300 FORI=1194TO1471:READQ:S4=S4+Q:POKE2
048+I,Q:NEXT
1310 IFS4<>33923THENPRINT"FEHLER IN DEN
DATAS DER ZEILEN 710-850!":END
1400 PRINT"ALLE DATAS SIND RICHTIG!
1410 PRINT:PRINT"NUN GEBEN SIE BITTE DIE
 FOLGENDEN          BEFEHLE EIN:
1420 PRINT:PRINT"POKE44,8:POKE46,13:POKE
45,192:CLR
1430 PRINT:PRINT:PRINT"DANACH KOENNEN SI
E DAS MASCHINEN-
1440 PRINT"PROGRAMM AUF KASSETTE ODER DI
SKETTE
1450 PRINT"ABSPEICHERN UND BEI BEDARF WI
E JEDES
1460 PRINT"ANDERE PROGRAMM LADEN UND STA
RTEN.
```

Listing »Funktionenplot« (Schluß)

BLOCKGROESSE 50

ZEILE	ANZAHL	SUMME	KEIN POKE?
120	50	2725	
140	100	9455	
170	150	16634	
200	200	22389	
220	250	27253	
250	300	32141	
270	350	37162	
300	400	42884	
320	450	49592	
350	500	57742	
380	550	63773	
400	600	69584	
430	650	75244	
450	700	80703	
480	750	86013	
500	800	91504	
530	850	97933	
550	900	100606	
580	950	106614	
610	1000	113226	
630	1050	120216	
660	1100	124605	
680	1150	129256	
710	1200	135005	
730	1250	140328	
760	1300	146150	
790	1350	152333	
810	1400	158304	
840	1450	165049	
GESAMT	1472	168259	

Das verwendete Programm zur Erstellung dieser Prüfsummenliste finden Sie in Ausgabe 9/84, Seite 67

Die Prüfsummenliste für »Funktionenplot«

wählt werden. Reicht eine grobe Darstellung, so wählt man den Parameter n größer. Läßt man den STEP-Befehl aus, wird n=0.03 gesetzt, was sich meist als bester Wert erweist. Die Anwendung des FPLOT-Befehls an einem Beispiel: Die Funktion f(x)=SQR(X) soll ausgegeben werden.

```
10 HIRES
20 SCNCLR
30 AXES160,100
40 FPLOTSQR(X)
```

Das Programm schaltet die hochauflösende Grafik ein, löscht den Grafikbildschirm, zeichnet die Koordinatenachsen und plottet die Funktion. Alle neuen Befehle des Programms können übrigens wie die normalen Basic-Befehle über die Shift-Taste abgekürzt werden.

Zum Schluß noch ein Tip: Es gibt Funktionen, die in der Hauptsache in einem Bereich verlaufen, der nicht innerhalb der Koordinaten liegt, die für den FPLOT-Befehl vorgesehen sind. In diesem Fall wird einfach auf den PLOT-Befehl zurückgegriffen. Ein kleines Basic-Programm plottet unter Verwendung dieses Befehls auch solche Funktionen problemlos. In diesem Fall ist außerdem die Benutzung des ONERROR-Befehls sinnvoll.

Hier noch Hinweise zum Eintippen und Abspeichern beziehungsweise Laden des Programms. Vor dem Eingeben des Basic-Quellprogramms werden die folgenden Befehle eingegeben, die den Anfang des Basic-Speicherbereichs nach oben verschieben:

POKE44,16:POKE4096,0:NEW

Danach gibt man das Quellprogramm ein. Es enthält die DATAs für das Maschinenprogramm sowie eine Prüfsummenroutine. Nachdem das Maschinenprogramm erzeugt ist, wird es abgespeichert. In Zukunft muß nur noch das Maschinenprogramm geladen und gestartet werden.

(Uwe Seimet/rg)

Supergrafik II

Dieses Programm liefert eine Grafik-Auflösung von 200 mal 256 Punkten und nützt damit den Bildschirm vollständig aus, und dies in der Grundversion des VC 20.

Es handelt sich hier um eine Weiterentwicklung des im 64'er, Ausgabe Mai 1984, Seite 81 abgedruckten Programms. Nach Eingabe der Größe des Koordinatensystems generiert das Programm den Grafen einer Funktion und ihrer Ableitung in verschiedenen Farben. Es ist sicher für viele Besitzer des VC 20, insbesondere für Schüler und Lehrer interessant.

Die Funktionsgleichung wird in Zeile 1 definiert. Das Programm legt den Koordinatenursprung in die Bildschirmmitte und zeichnet den Grafen der Funktion (rot) und ihrer Ableitung (gelb). Nach dem Programmstart wird der Definitionsbereich der Funktion abgefragt (—XMAX≦X≦XMAX) und der Maßstab für die Hochwert-Achse gewählt (—YMAX≦Y≦YMAX). Das Programm kann aus Gründen des beschränkten Speicherplatzes zwar »nur« 159 Felder aus jeweils 8 mal 16 Punkten ansprechen, was aber für diese Anwendung vollkommen ausreicht. Um Speicherplatz zu sparen, wurde auf Kommentare im Programm und Grundsätze zur übersichtlichen Programmierung bewußt verzichtet. (Rudolf Dörr/ev)

```
1 DEFFNF(X)=SIN(X)
2 PRINT"▒**FUNKTION+ABLEITUNG**":PRINT"▒
R.DOERR":PRINT"GUNZENHAUSEN,30.3.1984"
8 PRINT"▒▒▒▒▒FUNKTION IN ZEILE 1   DEFI
NIEREN!▒▒▒▒"
15 INPUT"XMAX";XM:INPUT"YMAX";YM
20 P=36864:POKEP+3,161:POKE55,0:POKE56,2
0:POKEP+1,18:POKEP,10:POKEP+2,153:PRINT"
▒"
25 POKE0,10:POKE1,0:FORI=828TO846:READP:
POKEI,P:NEXT:POKE2,20:SYS828
30 POKE36869,253:FA=6
90 Y=128:FORX=96TO102:GOSUB4000:NEXT:X=9
9:FORY=125TO131:GOSUB4000:NEXT
97 FA=2:D=0:SH=0.000001:H=.0001
100 FORX=0TO199
110 XW=XM*(2*X/199-1):YW=(FNF(XW+D*H+SH)
-D*FNF(XW+SH))/((H-1)*D+1):Y=(1-YW/YM)*1
27.5
115 IFY<0ORY>255.4THEN130
120 GOSUB4000
130 NEXT
140 IFD=1THEN200
150 FA=7:D=1:GOTO100
200 GETA$:IFA$=""THEN200
210 END
4000 :
4010 Y=INT(Y+.5):X=INT(X+.5):XS=INT(X/8)
:YS=INT(Y/16)
4040 BY=Y-YS*16:BI=7-X+XS*8:Z=7680+XS+25
*YS:ZF=Z+30720
4050 IFPEEK(Z)=32THENBC=BC+1:POKEZ,BC:PO
KEZF,FA
4055 IFBC>158THEN200
4060 BS=PEEK(Z):B=5120+16*BS+BY:P=PEEK(B
):P=POR2↑BI:POKEB,P:RETURN
5000 DATA162,0,169,0,160,0,145,1,200,208
,251,230,2,232,228,0,208,242,96
READY.
```

Listing »Supergrafik II«

Zeile	
1	Funktionsgleichung
15	Länge des positiven Teils der X-Achse (XMAX) und der Y-Achse (YMAX)
20	Ändern einiger Speicherinhalte der VIC-Kontrollregister zur Vergrößerung des Bildschirmfensters und Herabsetzen eines Zeigers (Ende Basic) zum Schutz des Zeichenspeichers im Hauptspeicher.
25	Einlesen und Ausführen eines kleinen Maschinenprogramms zur Löschung des Bereichs für die neuen Zeichen.
30	Verlegen des Zeichenspeichers in den Hauptspeicher, Farbe (blau) für den Koordinatenursprung.
90	Zeichnen eines kleinen Achsenkreuzes in der Bildschirmmitte.
97	Farbe (rot) für den Funktionsgraphen, Auswahlvariable D (0: Funktion, 1: Ableitung) »Sicherheitsvariable« SH (um Divisionen mit dem Divisor Null vorzubeugen) Schrittweite H zur Berechnung des Differenzquotienten
100-130	Ermitteln der Koordinaten der Punkte des Grafen (XW,YW) und Umrechnen auf Bildschirmkoordinaten (X,Y).
150	Umschalten auf die Ableitungsfunktion (D=1) und Farbe gelb
200	Wartestellung am Ende der Programmausführung
4000-4060	Eigentliches Grafik-Unterprogramm, Berechnung der nötigen Speicherinhalte im Zeichensatz und Ausgabe der Zeichen auf dem Bildschirm
5000	Speicherzelleninhalte für das Maschinenprogramm

Programmaufbau von »Supergrafik II«



Druckfehlerteufelchen

Kudiplo auch für den C 64

In Zeile 185 wurden zwei Ziffern vertauscht. Statt »... FOR I = 823 ...« muß es richtig heißen: »... FOR I = 832 ...«.

Test 40/80-Zeichenkarte für den VC 20, 10/84, Seite 20

Als Anbieter der Karte wurde versehentlich die falsche Firma genannt. Die richtige Bezugsadresse lautet:
Roßmöller GmbH
Finkenweg 1
5309 Meckenheim

Epedemic, 10/84, Seite 112

Hier hat unser Druckfehlerteufelchen wieder voll zugeschlagen und einen größeren Teil des Listings klammheimlich beiseitegeschafft. Es fehlen nämlich die Zeilen 50540 bis 58020.

Durch gutes Zureden konnte er allerdings in dieser Ausgabe auf Seite 94 zur Herausgabe des fehlenden Teils überredet werden.



Sprites ohne Esoterik

Auch fortgeschrittenen Programmierern bleibt es meist rätselhaft, wie der C 64 die Sprites auf den Bildschirm zaubert. Wir wollen dieses Thema einmal ohne Geheimniskrämerei (Esoterik) angehen.

Noch vor fünf Jahren war alles ganz einfach. Im guten alten PET 2001 wurde das Bild durch ein mittleres TTL-Bergwerk erzeugt. In den Nachfolgern CBM 30xx und so weiter wird diese Aufgabe durch einige TTL-ICs und den Bildschirmcontroller MC6845 erledigt. Bis dahin war alles überschaubar.

Da Commodore die Halbleiterfirma MOS Technology besitzt, liegt es nahe, eigene Video-Controller zu entwickeln. So treibt im VC 20 der noch relativ einfache VIC I (Video-Interface-Chip I) sein Unwesen. Im C 64 wirft der sehr komplexe VIC II die »Flammenschrift« auf den Bildschirm.

Die Funktionsweise dieses Video-Controllers soll dieser Bericht ein wenig enträtseln. Leider muß man es größtenteils durch Überlegung lösen.

Zunächst ein paar Worte zum Bildschirmformat: Es entspricht weitgehend der normalen Fernsehnorm. Je 312 Rasterzeilen ergeben fünfzig Bilder in der Sekunde. Jede Rasterzeile (Bild 1) ist 64 µs lang und beginnt mit einem Synchronisationsimpuls, der dem Monitor mitteilt, wann er mit der neuen Zeile beginnen soll. Ohne diese Synchronisation gäbe es nur Bildsalat und durchlaufende Bilder. Der Synchronisationsimpuls entspricht in der Helligkeitsskala der Videosignale einem Dunkelschwarz, so daß man den Strahlrücklauf nicht sehen kann. Es gibt übrigens nur fünf Helligkeitsstufen:

1. Schwarz
2. Rot, Blau, Braun, Grau 1
3. Violett, Grün, Orange, Hellrot, Grau 2, Hellblau
4. Türkis, Gelb, Hellgrün, Grau 3
5. Weiß

Es ist empfehlenswert, für die Vorder- und Hintergrundfarbe Farben aus verschiedenen Helligkeitsstufen zu wählen, da man sonst auf farblosen Monitoren nicht viel sehen kann...

40 µs der Rasterzeile werden für das eigentliche Bild benützt. Bei der Taktfrequenz (des VIC) von zirka 8 MHz ergeben sich 512 Punkte auf der ganzen Rasterzeile, was mit der horizontalen Auflösung der Positionen der Sprites übereinstimmt. Aus den 64µs ergibt sich eine Zeilenfrequenz von 15625 Hz, die für das unangenehme Pfeifen verantwortlich ist, das man bei jedem Bildschirm hören kann, sofern man noch gute Ohren hat. Das horizontale Scrolling ist übrigens sehr einfach zu verwirklichen. Statt dem ganzen Bild wird kurzerhand der Synchronisationsimpuls und das durch den Rand gebildete »Fenster« verschoben.

Das vertikale Format ist folgendermaßen aufgebaut:
4 Zeilen Schwarz, damit man den Strahlrücklauf nicht sieht
3 Zeilen Synchronisationsimpuls
4 Zeilen Schwarz
51 Zeilen Rand
200 Zeilen Bild
51 Zeilen Rand

Den aufmerksamen Lesern wird aufgefallen sein, daß ich hier von 312 Zeilen pro Bild rede und nicht von 625 Zeilen, wie es beim normalen Fernseher der Fall ist. Beim Fernseher werden nacheinander zwei »Halbbilder«, die zueinander um eine halbe Rasterzeile versetzt sind, geschrieben, so daß die Auflösung höher ist. Nachteilig ist bei diesem Verfahren die geringe Bildwiederholfrequenz von 25 Hz, bei der das Bild leicht flimmert. Beim C 64 sind beide Bilder identisch und nicht versetzt (non-interlace), die Bildwechselfrequenz beträt 50 Hz.

Um ein Zeichen darzustellen, muß der VIC in einer einzigen Mikrosekunde folgende Informationen lesen:
1. POKE-Wert des Zeichens aus der Videomatrix; zeigt in den Zeichengenerator
2. Farbe aus der Farbmatrix
3. Daten aus dem Zeichengenerator.

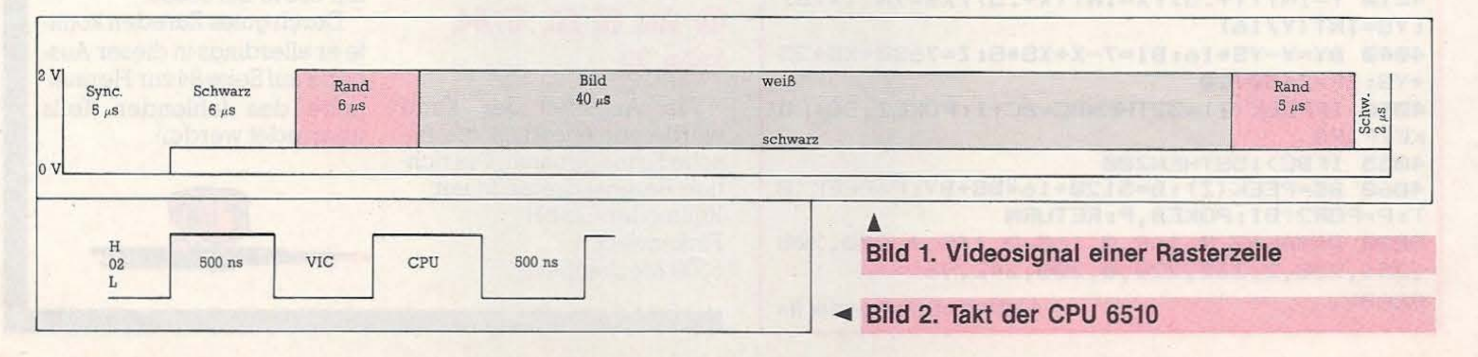
Das heißt, zusammen mit den Zugriffen der CPU müßte der Speicher mit einer Zyklusfrequenz von 4 MHz (!) betrieben werden. Das ist nicht möglich. Dieses Problem wird auf eine andere Weise gelöst.

Die CPU 6510 gibt an ihrem Ausgang Pin 2 einen Takt von zirka 1 MHz ab (Bild 2). Wenn Pin 2 Low ist, ist der Bus frei, wenn Pin 2 High ist, benötigt die CPU den Bus für ihre Speicherzugriffe.

Der VIC liest, während Pin 2 Low ist, die Daten aus dem Zeichengenerator oder aus der hochauflösenden Grafik.

Das Farb-RAM wird vom VIC parallel zum normalen RAM gelesen. Der VIC hat also einen 12-Bit-Datenbus.

Zum Lesen der Videomatrix ist keine Zeit mehr übrig. Deswegen muß der VIC regelmäßig die CPU anhalten, um die Daten lesen zu können. Durch dieses »Kaltstellen« wird der Prozessor natürlich verlangsamt. Damit die CPU nur um etwa zehn Prozent und nicht um die Hälfte verlangsamt wird, hat der VIC intern Puffer für die Video- und Farbdaten der aktuellen Zeile.



▲ **Bild 1. Videosignal einer Rasterzeile**

◄ **Bild 2. Takt der CPU 6510**

	a) sichtbarer Teil (40 µs)	Rand, Austastlücke (24 µs)	02
a)	VIC: Daten aus Zeichensatz/Grafik	Sprite-Daten 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7	L
	VIC: Daten aus Farb- und Videomatrix	Positionen der Spriteblöcke?	H
b)	VIC: Daten aus Zeichensatz/Grafik	Sprite-Daten 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7	L
	CPU	CPU	H

Bild 3. Zeitliche Anordnung der Speicherzugriffe

Daraus ergibt sich die auf Bild 3 gezeigte Reihenfolge der Zugriffe:

a) Letzte Rasterzeile der vorherigen Zeile: Die CPU wird angehalten, in die Puffer werden die Daten für die neue Zeile gelesen.

b) Während den sieben ersten Rasterzeilen der neuen Zeile wird die CPU nicht angehalten.

a) Wieder von vorn.

Leider hat auch dieses Verfahren Nachteile. Erstens wird der Prozessor verlangsamt, zweitens geraten bei zeitkritischen Programmen durch das Anhalten des Prozessors die Zeitverhältnisse aus dem Lot. Deswegen wird zum Beispiel beim Laden von Programmen von Kassette der Bildschrim abgeschaltet!

Und nun zu den Sprites. Unbegrenztes Vertrauen in die Leistungsfähigkeit der CPU 6510 ist nicht angebracht. Im VIC ist kein Maschinenprogramm versteckt. Es wäre ganz einfach zu langsam. Statt dessen werden die Sprites durch aufwendige Hardware erzeugt. Dieser Aufwand macht aber nicht soviel aus, da eben alles in einem Chip versammelt ist. Schwierig wird die Entwicklung eines solchen Video-Controllers vor allem durch die notwendige hohe Geschwindigkeit: Der IC hat immerhin eine Taktfrequenz von 8 MHz zu verkraften. Deswegen wird der VIC gewaltig heiß und residiert in einem unter kühlendem Blech verborgenen Keramikgehäuse.

Trotz allem ist die Logik für die Sprites eigentlich verblüffend einfach und elegant. Da Digitalelektroniker auch beim Einschlafen noch Gatter zählen, die über Schafe springen, und auch sonst mit jedem Gatter geizen (einfache ICs sind billiger), kann man schließen, daß wahrscheinlich auch hier der einfachste Weg benützt wird.

Die Daten der Sprites müssen zuerst gelesen werden. Die Logik zur Errechnung der Adressen der Sprites soll hier nicht erklärt werden, da sie nicht besonders interessant ist. Es ist mir leider nicht bekannt, wann die Spritezeiger (am Ende der Videomatrix) gelesen werden.

In jeder Rasterzeile müssen die Daten aller Sprites gelesen werden, also 3 x 8 Byte, die nahtlos in das »gemütliche Eckchen« in der Austastlücke passen. Jetzt wissen wir auch, wieso die Sprites ein so unmögliches Format (24 x 21 Pixel) haben...

Ich möchte nun anhand von Bild 4 erklären, wie die Sprites dort angezeigt werden, wo sie hingehören. Im Schema ist nur die Schaltung für ein Sprite gezeichnet, die anderen Sprites sind gleichartig aufgebaut.

Die Spritedaten werden in einen Puffer gelesen. Ein Zähler gibt die Nummer der aktuellen Rasterspalte, also die X-Koordinate, an. Davon wird die X-Koordinate der Sprites abgezogen, und man erhält eine auf den »Ursprung« des Sprites bezogene Koordinate. Die Ausdehnung von Sprites in der X-Richtung ist sehr einfach: Die Koordinate wird einfach durch zwei geteilt (in Binärsystem sehr einfach: rechts schieben). Das gleiche geschieht übrigens auch bei der Ausdehnung von Sprites in der Y-Richtung. Ein Multiplexer gibt das, durch die so erhaltene Koordinate, gewählte Bit (oder Bitpaar bei mehrfarbigen Sprites) aus. Falls die Koordinate nicht im Bereich des Sprites liegt, gibt der Multiplexer einfach den Wert für »Sprite transparent« — also 0 — aus.

Wie werden die Sprites nach der Priorität geordnet und Kollisionen von Sprites mit anderen Sprites oder mit dem Vordergrund des normalen Bilds erkannt?

Jedes Sprite gibt ein Signal von sich, das angibt, ob das Sprite jetzt transparent oder »deckend« ist. Eine relativ einfache Schaltung (in TTL nur 28 Gatter: 74LS148) erzeugt die Nummer des Sprites mit der höchsten Priorität, das gerade deckend ist oder zeigt an, daß gar kein Sprite deckend ist.

Nun muß noch entschieden werden, wer jetzt Vorfahrt hat (in der Reihenfolge der Prioritäten): Der Rand, Sprites im Vordergrund, das normale Bild, Sprites im Hintergrund oder der Hintergrund. Entsprechend dieser Entscheidung wird der richtige Farbcode ausgewählt und an den PAL-Codierer weitergegeben, der das Helligkeitssignal (Videosignal) und das Farbsignal (Chroma) erzeugt.

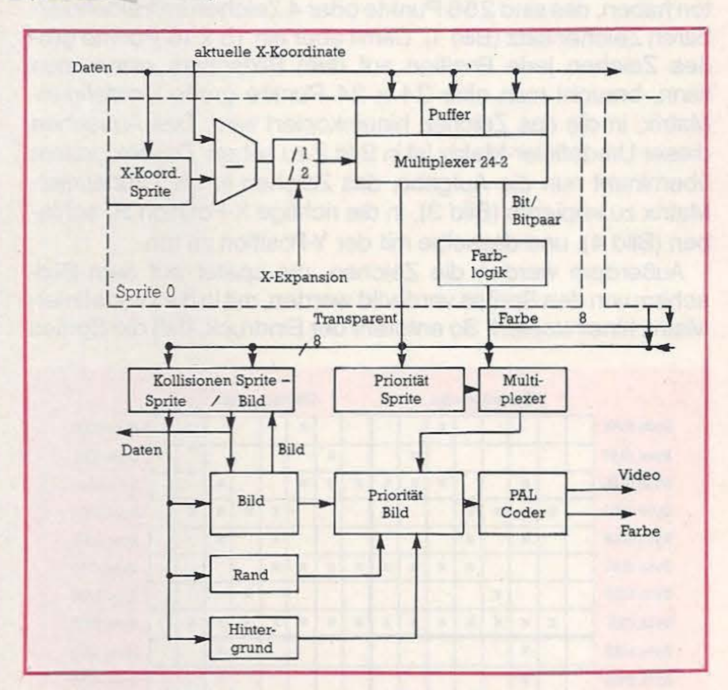


Bild 4. Schema der Sprite-/Bildlogik (stark vereinfacht)

Die Erkennung von Kollisionen ist keine schwierige Sache mehr. Wenn sowohl das Bild als auch ein oder mehrere Sprites nicht transparent sind, werden im Kollisionregister für Sprite-Bild-Kollisionen die Bits der nicht transparenten Sprites gesetzt.

Sprite-Sprite-Kollisionen sind etwas schwieriger auszuwerten, auch hier bleibt der Aufwand aber im Rahmen.

Ich hoffe, daß ich bei den Lesern mit diesem Artikel jeglichen Geisterglauben ausgetrieben habe. (Pascal Dornier/aa)

Pseudo-Sprites auf dem VC 20

Der VC 20 kennt von Haus aus leider nicht die freibeweglichen Grafikobjekte des C 64, die sogenannten Sprites. Das bedeutet aber nicht, daß man auf die Vorteile der Sprites oder MOBs gänzlich verzichten muß.

Das Programm ist für den VC 20 mit 8 KByte Speichererweiterung konzipiert. Es läuft jedoch mit einigen Änderungen auch bei nur 3 KByte Speichererweiterung.

Vor dem Eintippen oder Laden muß man POKE 44,32:POKE 8192,0: NEW eingeben, womit der Basic-Anfang im Speicher auf die Adresse dezimal 8193 ($2001) erhöht wird. Somit ergibt sich folgende Speicheraufteilung:

4096 — 4607 Bildschirm
4608 — 8191 frei
8192 — 16383 (bei + 8 KByte) Basic-Programmspeicher
8192 — 24575 (bei + 16 KByte) Basic-Programmspeicher
8192 — 32767 (bei + 24 KByte) Basic-Programmspeicher

Der freie Bereich wird nun vollständig von dem Maschinenspracheprogramm gebraucht. Die Aufteilung des Speicherraums ist die folgende:

4608 — 5119	Sprite-Control-Block (SCB), wird später erklärt
5120 — 6143	freidefinierbarer Zeichensatz, die ersten 128 Zeichen, stehen zur freien Verfügung
6144 — 7167	freidefinierbarer Zeichensatz, die zweiten 128 Zeichen, werden vom Programm zur Erstellung der 9 Sprites gebraucht und stehen somit nicht zur freien Verfügung
7168 — 8191	Maschinenspracheprogramm, Beschreibung siehe Text

Die Pseudo-Sprites sollten eine Auflösung von 16 x 16 Punkten haben, das sind 256 Punkte oder 4 Zeichen im freidefinierbaren Zeichensatz (Bild 1). Damit aber ein 16 x 16 Punkte großes Zeichen jede Position auf dem Bildschirm einnehmen kann, braucht man eine 24 x 24 Punkte große Umdefinier-Matrix, in die das Zeichen hineinkopiert wird. Das Aussehen dieser Umdefinier-Matrix ist in Bild 2 zu sehen. Das Programm übernimmt nun die Aufgabe, das Zeichen in die Umdefinier-Matrix zu kopieren (Bild 3), in die richtige X-Position zu schieben (Bild 4), und dasselbe mit der Y-Position zu tun.

Außerdem werden die Zeichen, die später auf dem Bildschirm von den Sprites verdeckt werden, mit in die Umdefinier-Matrix hineinkopiert. So entsteht der Eindruck, daß die Sprites wirklich über die Zeichen wandern. Beim späteren Löschen werden die verdeckten Zeichen wieder hergestellt. Wie funktioniert das nun?

Im Speicher ab dezimal 4608 ist 9mal (für jedes Sprite einer) der sogenannte Sprite-Control-Block (SCB) eingerichtet. Er hat die Aufgabe, die momentane X- und Y-Position, die Farbe des Sprites, den Bildschirmmodus (gesetzt/gelöscht) des Sprites, die durch die Umdefinier-Matrix verdeckten 9 Zeichen und Farben zwischenzuspeichern:

Byte 0 — 8	Bildschirmcode der verdeckten Zeichen
Byte 9 — 17	Farbcode der verdeckten Zeichen
Byte 18	X-Position des Sprites
Byte 19	Y-Position des Sprites
Byte 20	Farbe des Sprites
Byte 21	Bildschirmmodus (gesetzt= $00/gelöscht=$FF)

Die Basisadresse des SCB errechnet sich somit aus der Formel Basisadresse = 4608 + Spritenummer x 22. Das Zwischenspeichern und die Auswertung der Parameter übernimmt natürlich das Maschinenprogramm. Über den SCB werden auch im nachfolgend beschriebenen Programm »Sprite-Definer« die Sprites initialisiert und deren Farbe festgelegt. Da nur die oberen 128 Zeichen des Zeichensatzes für die Sprites verwendet werden, hat man eine ausreichende Anzahl von noch frei definierbaren Zeichen, nämlich genau 128, zur Verfügung. Außerdem kommen jeweils 13 Zeichen, nämlich 4 für das Sprite und 9 für die Umdefinier-Matrix hinzu, wenn man auf ein Sprite verzichtet. Die Matrixen werden im Speicher so abgelegt:

128 — 131	Grundmatrix Sprite 0
132 — 140	Umdef.-Matrix Sprite 0
141 — 144	Grundmatrix Sprite 1
145 — 153	Umdef.-Matrix Sprite 1

Konkret wird das Programm (Listing 1) nun folgendermaßen bedient: Vor dem Laden oder Eingeben wird POKE 44,32:POKE 8192,0:NEW eingetippt. Ist nun das Maschinen-

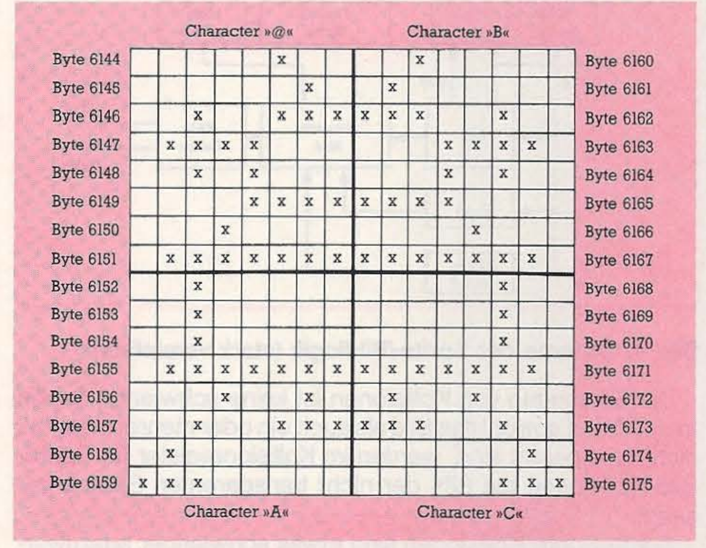


Bild 1. Die Speicheraufteilung in der Grundmatrix des Sprites Nummer 0. Das höchstwertige Bit eines Bytes steht links.

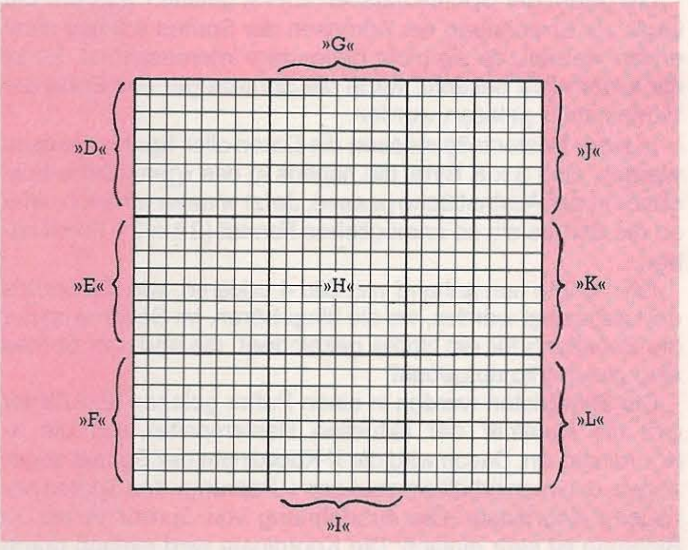


Bild 2. Die Undefinier-Matrix hat 24 x 24 Punkte und ist notwendig, um Sprites auch punktweise verschieben zu können



spracheprogramm im Speicher, kann es mit einem Monitorprogramm auch noch einmal abgespeichert werden. Später muß man es nur noch mit LOAD »name«, 1,1 laden.

Die Bedienung:
Sind die Sprites definiert, muß dem Maschinenspracheprogramm mitgeteilt werden, wo der Zeichensatz liegt, den es verwalten soll. Das geschieht mit den Befehlen POKE 677, Lowbyte:POKE 678, Highbyte, in unserem Fall also POKE 677,0:POKE 678,20, da der Zeichensatz auf der Adresse 5120 beginnt.

Soll nun ein Sprite auf den Bildschirm, muß zuerst einmal in Adresse 683 die Spritenummer gePOKEt werden (Achtung, keine Zahl über 8 angeben, da sich das Programm dann selbst zerstören könnte). Schließlich werden in Adresse 673 die X-Koordinate (maximal 159) und 674 die Y-Koordinate (maximal 167) gesetzt. Dann kann das Programm mit SYS 8021 sofort aufgerufen und auf dem Bildschirm das Sprite betrachtet werden, vorausgesetzt man hat vorher mit POKE 36869,205 auf den freidefinierbaren Zeichensatz geschaltet.

Wird nun das Sprite auf eine andere Position gesetzt, so verschwindet es vollständig von der alten Position, und die Zeichen, die auf diesem Platz waren, erscheinen wieder mit ihrer alten Farbe. Will man aber das Sprite ganz vom Bildschirm löschen, POKEt man wieder in 683 die Spritenummer und ruft das Maschinenspracheprogramm diesmal mit SYS 8099 auf. PRINT »CLR/HOME« sollte man nicht verwenden, da im SCB noch die alten Bildschirmzeichen gespeichert sind und beim nächsten Setzen wieder auf ihren alten Plätzen auf dem Bildschirm erscheinen würden.

Der Sprite-Generator

Nun zum Programm »Sprite-Definer« (Listing 2).

Dieses Programm ist ein Sprite-Generator in Basic, der bei der Erstellung von Sprites recht hilfreich sein kann. Das Programm verdeutlicht auch, wie die Definition der Sprites und die Bedienung des Maschinenspracheprogramms erfolgt.

Obwohl sich das Programm fast von selbst erklärt, hier doch einige kurze Erläuterungen:

Startet man das Programm mit RUN, erscheint als erstes die Begrüßung und die Aufforderung »Bitte warten!«. Das Programm kopiert nämlich jetzt den Zeichensatz aus dem ROM ins RAM, was in Basic naturgemäß etwas dauert.

Jedesmal, wenn man in einem Menüteil eine Eingabe gemacht hat, wird man »Richtig?« gefragt. Tippt man hier für N (Nein), so kann die Eingabe wiederholt werden. Drückt man aber den Linkspfeil, so kommt man wieder ins Hauptmenü.

Die Tastenbelegung im Editiermodus:

Cursor-Tasten	Cursor-Bewegungen
Leertaste	Punkt setzen
Delete-Taste	Punkt löschen
CLR/HOME	Gitter löschen
RETURN	Modus beenden mit Änderung des Sprites, vorher aber Abfrage
Linkspfeil	Modus beenden, aber ohne Änderung des Sprites
I-Taste	Sprite invertieren

Bei der Funktion »Weiter« kommt man in ein zweites Menü, das weitere Funktionen zur Verfügung stellt. Aus diesem Menü gelangt man mit »zurück« wieder ins Hauptmenü. Beim Speichern werden die Sprites als reiner Speicherauszug auf Kassette gebracht, so daß das Laden im Prinzip auch mit LOAD »name«,1,1 möglich ist.

Sicherlich kann das Maschinenspracheprogramm noch weiter verbessert werden. So wäre zum Beispiel eine Spritesteuerung per Interrupt durchaus denkbar. Leider funktioniert das Maschinenprogramm nicht mit den üblichen Grafikmodulen, da diese den Bildschirminhalt auch mit dem freidefinierbaren Zeichensatz aufbauen. Sollen Sprites auch miteinander oder übereinander dargestellt werden, dann muß das Setzen und Löschen nach folgender Reihenfolge durchgeführt werden, da es sonst zu Schwierigkeiten mit dem SCB kommen kann:

Sprite 0 setzen, Sprite 1 setzen,..., Sprite n setzen. Hiernach die Berechnungen für die neuen Positionen durchführen.
Sprite n löschen, Sprite n-1 löschen,..., Sprite 0 löschen. Danach Vorgang von oben wiederholen.

Noch eins zum »Sprite-Definer«: Die erste REM-Zeile muß auf jeden Fall mit 16 Sternchen eingegeben werden, da sich das Programm später mit POKEs selbst verändert und andernfalls, wäre die REM-Zeile kürzer, die folgende Zeile in Mitleidenschaft ziehen würde. Doch nun wünsche ich allen, die das Programm eintippen, viel Spaß und vielleicht ein bißchen C 64-Feeling.

(Markus Leberecht/ev)

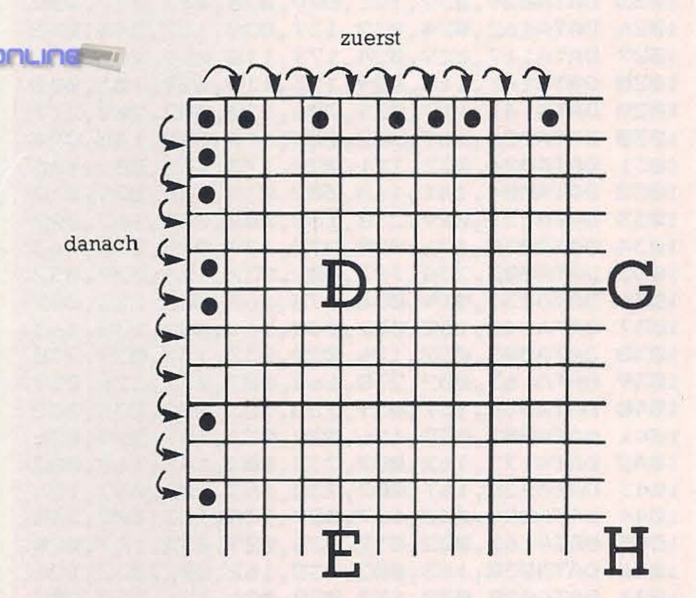


Bild 4. Zur punktgenauen Justierung wird das Sprite nach dem Kopieren in die Umdefinier-Matrix noch horizontal und vertikal verschoben

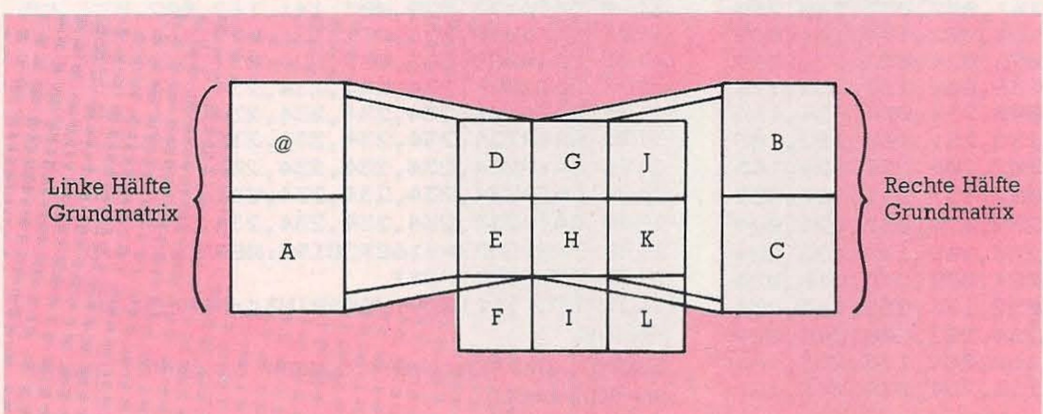


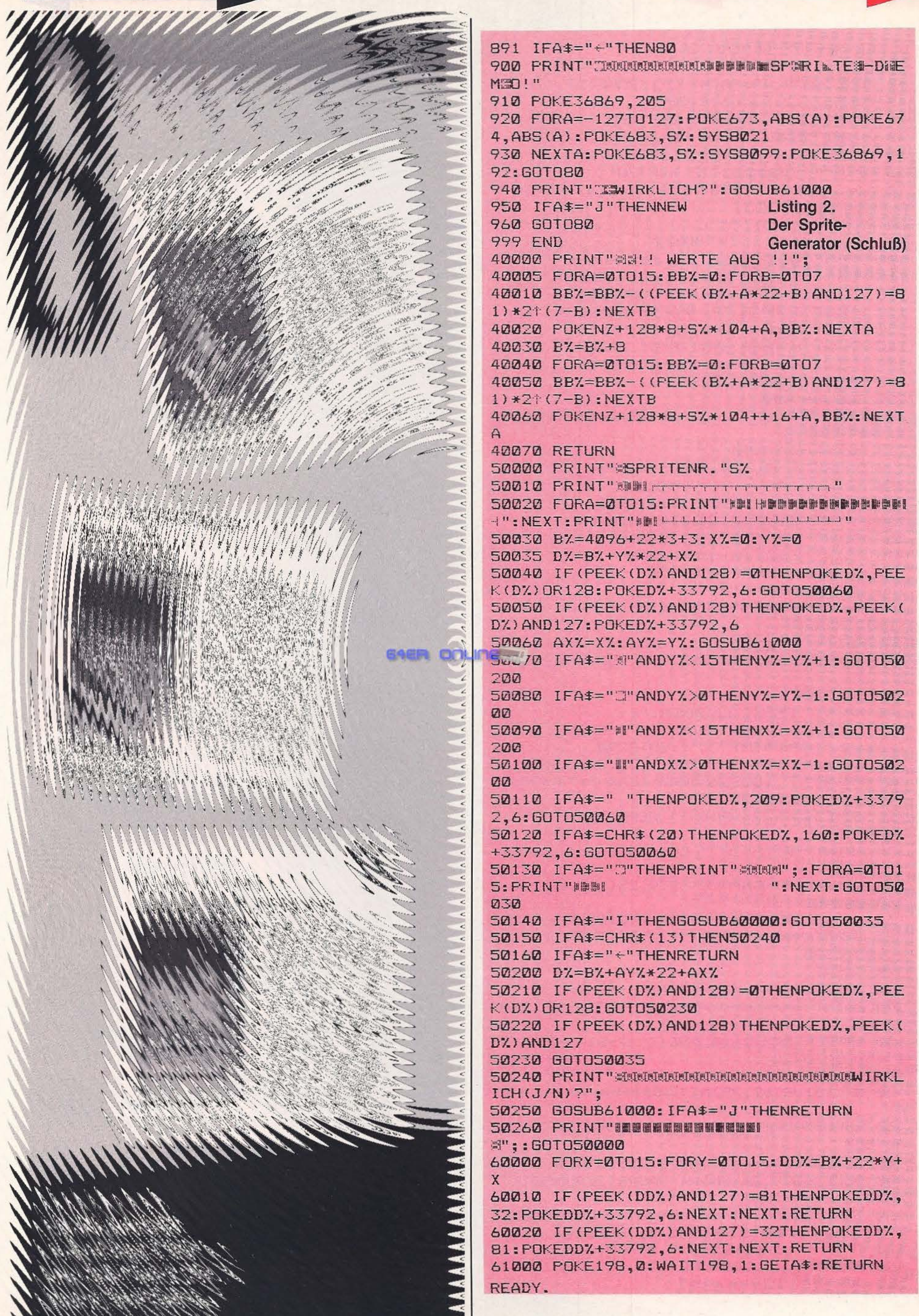
Bild 3. Hineinkopieren der Grundmatrix in die Umdefinier-Matrix

```
1 REM PSEUDOSPRITES
2 REM FUER VC 20
3 REM (DATA-LADER)
4 REM
5 REM SPRITES VON 7168 BIS 8192
6 REM
7 REM 'SAVE' VOR 'RUN' !
8 REM
1001 DATA024,032,121,029,173,165,002,133
1002 DATA025,173,166,002,133,026,142,169
1003 DATA002,173,163,002,042,046,169,002
1004 DATA024,042,046,169,002,024,042,046
1005 DATA169,002,024,101,025,133,025,165
1006 DATA026,109,169,002,133,026,024,165
1007 DATA025,105,032,133,029,165,026,105
1008 DATA000,133,030,173,161,002,041,007
1009 DATA141,167,002,173,161,002,024,074
1010 DATA074,074,141,161,002,024,173,162
1011 DATA002,041,007,141,168,002,173,162
1012 DATA002,074,074,074,141,162,002,032
1013 DATA121,029,162,072,145,029,200,202
1014 DATA208,250,024,160,000,024,165,029
1015 DATA109,168,002,133,031,165,030,105
1016 DATA000,133,032,177,025,145,031,200
1017 DATA192,016,208,247,024,165,031,105
1018 DATA024,133,031,165,032,105,000,133
1019 DATA032,024,165,025,105,016,133,027
1020 DATA165,026,105,000,133,028,160,000
1021 DATA024,177,027,145,031,200,192,016
1022 DATA208,247,173,167,002,201,000,240
1023 DATA060,234,169,024,141,170,002,165
1024 DATA029,141,118,029,165,030,141,119
1025 DATA029,032,121,029,024,032,117,029
1026 DATA162,024,032,117,029,162,048,032
1027 DATA117,029,024,173,118,029,105,001
1028 DATA141,118,029,173,119,029,105,000
1029 DATA141,119,029,206,170,002,208,217
1030 DATA206,167,002,208,197,032,148,029
1031 DATA024,032,121,029,173,163,002,105
1032 DATA004,141,163,002,032,126,029,032
1033 DATA137,029,238,163,002,238,162,002
1034 DATA032,126,029,032,137,029,238,163
1035 DATA002,238,162,002,032,126,029,032
1036 DATA137,029,056,173,162,002,233,002
1037 DATA141,162,002,238,161,002,238,163
1038 DATA002,032,126,029,032,137,029,238
1039 DATA163,002,238,162,002,032,126,029
1040 DATA032,137,029,238,163,002,238,162
1041 DATA002,032,126,029,032,137,029,056
1042 DATA173,162,002,233,002,141,162,002
1043 DATA238,161,002,238,163,002,032,126
1044 DATA029,032,137,029,238,163,002,238
1045 DATA162,002,032,126,029,032,137,029
1046 DATA238,163,002,238,162,002,032,126
1047 DATA029,032,137,029,096,126,120,007
1048 DATA096,169,000,170,168,096,024,174
1049 DATA162,002,172,161,002,032,240,255
1050 DATA096,024,173,163,002,174,164,002
1051 DATA032,161,234,096,024,032,121,029
1052 DATA133,251,173,136,002,133,252,174
1053 DATA162,002,224,000,240,016,024,165
1054 DATA251,105,022,133,251,165,252,105
1055 DATA000,133,252,202,208,240,024,165
1056 DATA251,109,161,002,133,251,165,252
1057 DATA105,000,133,252,024,032,121,029
1058 DATA169,000,133,253,169,148,133,254
1059 DATA174,162,002,224,000,240,016,024
1060 DATA165,253,105,022,133,253,165,254
1061 DATA105,000,133,254,202,208,240,024
1062 DATA165,253,109,161,002,133,253,165
1063 DATA254,105,000,133,254,024,032,121
1064 DATA029,165,029,133,031,165,030,133
1065 DATA032,024,032,121,029,162,000,173
1066 DATA165,002,133,027,173,166,002,133
1067 DATA028,189,098,030,048,075,168,177
1068 DATA251,157,060,003,072,169,000,141
1069 DATA169,002,104,010,046,169,002,010
1070 DATA046,169,002,010,046,169,002,101
1071 DATA027,133,027,173,169,002,101,028
1072 DATA133,028,160,000,177,027,017,031
1073 DATA145,031,200,192,008,208,245,024
1074 DATA165,031,105,008,133,031,165,032
1075 DATA105,000,133,032,189,098,030,168
1076 DATA177,253,157,069,003,232,076,007
1077 DATA030,096,000,022,044,001,023,045
1078 DATA002,024,046,255,024,032,121,029
1079 DATA173,136,002,133,252,152,133,251
1080 DATA169,000,133,253,169,148,133,254
1081 DATA173,162,002,240,017,168,024,165
1082 DATA251,105,022,133,251,165,252,105
1083 DATA000,133,252,136,208,240,024,165
1084 DATA251,109,161,002,133,251,165,252
1085 DATA105,000,133,252,173,162,002,240
1086 DATA017,168,024,165,253,105,022,133
1087 DATA253,165,254,105,000,133,254,136
1088 DATA208,240,024,165,253,109,161,002
1089 DATA133,253,165,254,105,000,133,254
1090 DATA024,032,121,029,189,098,030,048
1091 DATA015,168,189,060,003,145,251,189
1092 DATA069,003,145,253,232,076,204,030
1093 DATA096,024,032,121,029,173,161,002
1094 DATA074,074,074,141,161,002,173,162
1095 DATA002,074,074,074,141,162,002,076
1096 DATA108,030,024,032,121,029,177,251
1097 DATA153,060,003,200,192,022,208,246
1098 DATA096,024,032,121,029,185,060,003
1099 DATA145,251,200,192,022,208,246,096
1100 DATA024,032,121,029,133,251,169,018
1101 DATA133,252,169,128,141,163,002,173
1102 DATA171,002,201,000,240,038,024,165
1103 DATA251,105,022,133,251,165,252,105
1104 DATA000,133,252,232,236,171,002,208
1105 DATA237,024,032,121,029,024,173,163
1106 DATA002,105,013,141,163,002,232,236
1107 DATA171,002,208,242,096,024,032,024
1108 DATA031,024,032,121,029,032,250,030
1109 DATA173,161,002,072,173,162,002,072
1110 DATA173,081,003,048,015,173,078,003
1111 DATA141,161,002,173,079,003,141,162
1112 DATA002,032,225,030,024,032,121,029
1113 DATA104,141,162,002,141,079,003,104
1114 DATA141,161,002,141,078,003,173,080
1115 DATA003,141,164,002,032,000,028,169
1116 DATA000,141,081,003,032,024,031,032
1117 DATA009,031,096,032,024,031,024,032
1118 DATA121,029,032,250,030,173,081,003
1119 DATA048,030,173,078,003,141,161,002
1120 DATA173,079,003,141,162,002,032,225
1121 DATA030,024,032,121,029,169,255,141
1122 DATA081,003,032,024,031,032,009,031
1123 DATA096,234,234,234,234,234,234,234
1124 DATA234,234,234,234,234,234,234,234
1125 DATA234,234,234,234,234,234,234,234
1126 DATA234,234,234,234,234,234,234,234
1127 DATA234,234,234,234,234,234,234,234
1128 DATA234,234,234,234,234,234,234,234
1130 S=0:FORI=7168TO8191:READD:S=S+D
1131 POKEI,D:NEXT
1132 IFS<>111729THENPRINT"■FEHLER!■"
READY.
```

Listing 1. Das Maschinenprogramm zur Sprite-Kontrolle als Basic-Lader

```
1 REM SPRITE-GENERATOR
2 REM
10 POKE36879,25:PRINT"";:POKE157,128
15 POKE677,0:POKE678,20
20 PRINT"    SPRITE-DEFINER    "
30 PRINT"    FUER VC-20    "
50 PRINT"   MARKUS LEBERECHT"
60 PRINT"BITTE WARTEN!"
70 AZ=32768:NZ=5120:FORA=0TO2047:POKENZ+
A,PEEK(AZ+A):NEXT
80 PRINTCHR$(8);"";
90 PRINT"       MENUE       "
100 PRINT"F1 SPRITE ERSTELLEN
110 PRINT"F2 SPRITE EDITIEREN
120 PRINT"F3 SPRITE LOESCHEN
130 PRINT"F4 SP.-FARBE AENDERN
150 PRINT"F5 SPRITE(S) SAVEN
160 PRINT"F6 SPRITE(S) LADEN
170 PRINT"F7 SPRITE KOPIEREN
180 PRINT"F8 WEITER
190 GOSUB61000
200 F$="":FORA=1TO8:IFMID$(F$,A,
1)=A$THEN220
210 NEXTA:GOTO190
220 ONAGOTO290,320,450,520,570,690,750
230 PRINT"        MENUE        "
240 PRINT"F1 VERIFIZIEREN
250 PRINT"F2 INITIALISIEREN
251 PRINT"F3 SPRITE-DEMO
260 PRINT"F7 ZURUECK
270 PRINT"F8 BEENDEN
280 GOSUB61000
285 F$="":FORA=1TO8:IFMID$(F$,A,
1)=A$THEN287
286 NEXTA:GOTO280
287 ONAGOTO810,840,880,280,280,280,80,94
0
288 GOTO280
290 PRINT"SPRITENUMMER:";
300 INPUTS%:PRINT"RICHTIG?":GOSUB61000:
IFA$="N"THENPRINT"";:GOTO290
302 IFA$="←"THENPOKE36879,25:GOTO80
305 PRINT"";:GOSUB50000:IFA$="←"THEN80
310 GOSUB40000:GOTO80
320 POKE36879,24:PRINT"SPRITENUMMER";:
INPUTS%
330 PRINT"RICHTIG?":GOSUB61000:IFA$="N"
THEN320
335 IFA$="←"THENPOKE36879,25:GOTO80
340 BA%=NZ+1024+S%*104
350 PRINT"";:FORA=0TO15:FORB=0TO7
360 IFPEEK(BA%+A)AND(2↑(7-B))THENPRINT"*
";:GOTO380
370 PRINT" ";
380 NEXTB:PRINT:PRINT"";:NEXTA
390 PRINT"";:FORA=0TO15:F
ORB=0TO7
400 IFPEEK(BA%+16+A)AND(2↑(7-B))THENPRIN
T"*";:GOTO420
410 PRINT" ";
420 NEXTB:PRINT:PRINT"";:NEXT
A
430 GOSUB50000:IFA$="←"THENPOKE36879,25:
GOTO80
440 GOSUB40000:POKE36879,25:GOTO80
450 POKE36879,26:PRINT"SPRITENUMMER:":
PRINT"(9=KEINS)
460 INPUTS%:PRINT"RICHTIG?":GOSUB61000
470 IFA$="N"OR(S%>9ORS%<0)THEN450
480 IFS%=9THENPOKE36879,25:GOTO80
490 PRINT"!! LOESCHE !!"
500 BA%=NZ+1024+S%*104
510 FORA=0TO31:POKEBA%+A,0:NEXT:POKE3687
9,25:GOTO80
520 POKE36879,27:PRINT"SPRITEFARBEN:"
530 FORA=0TO8:PRINT"SPRITE NR."A"---"PE
EK(4608+A*22+20)AND15:NEXTA
540 INPUT"WELCHES (9=EXIT)";S%:IFS%<0OR
S%>8THENPOKE36879,25:GOTO80
550 INPUT"FARBE";F%:IFF%<0ORF%>15THENPOK
E36879,25:GOTO80
560 POKE4608+22*S%+20,F%:GOTO520
570 POKE36879,28:INPUT"ANF.-SPRITE:";S
A%:IFSA%<0ORSA%>8THEN570
580 INPUT"ENDSPRITE:     ";SE%:IFSE%<0
ORSE%>8THENPRINT"";:GOTO580
585 PRINT"RICHTIG?":GOSUB61000:IFA$="N"
THEN570
587 IFA$="←"THENPOKE36879,25:GOTO80
590 IFSA%>SE%THEN570
600 ST%=NZ+1024+SA%*104:EN%=NZ+1024+(SE%
+1)*104
610 POKE185,3:INPUT"FILENAME:";FI$:FI$=
LEFT$(FI$,16):IFFI$=""THENPRINT"";:GOT
O610
620 FORA=1TOLEN(FI$):POKE8192+5+A,ASC(MI
D$(FI$,A,1)):NEXT:POKE183,LEN(FI$)
640 POKE186,1
650 POKE187,6:POKE188,32
660 POKE193,ST%-256*INT(ST%/256):POKE194
,ST%/256
670 POKE174,EN%-256*INT(EN%/256):POKE175
,EN%/256
680 SYS63106:POKE36879,25:GOTO80
690 POKE36879,29:INPUT"FILENAME:";FI$
700 PRINT"RICHTIG?":GOSUB61000:IFA$="N"
THEN690
705 IFA$="←"THENPOKE36879,25:GOTO80
710 IFLEN(FI$)=0THEN730
720 FORA=1TOLEN(FI$):POKE8192+5+A,ASC(MI
D$(FI$,A,1)):NEXT
730 POKE183,LEN(FI$):POKE186,1:POKE187,6
:POKE188,32
740 SYS62786:POKE36879,25:GOTO80
750 POKE36879,30:INPUT"QUELLE=";S1%:IF
S1%<0ORS1%>8THENPOKE36879,25:GOTO80
760 INPUT"ZIEL=";S2%:IFS2%<0ORS2%>8THEN
POKE36879,25:GOTO80
770 PRINT"RICHTIG?":GOSUB61000:IFA$="N"
THEN750
775 IFA$="←"THENPOKE36879,25:GOTO80
780 PRINT"!! KOPIERE !!"
790 S1%=NZ+1024+104*S1%:S2%=NZ+1024+104*
S2%
800 FORA=0TO31:POKES2%+A,PEEK(S1%+A):NEX
T:POKE36879,25:GOTO80
810 POKE36879,31:PRINT"BESTAETIGEN":GO
SUB61000
815 IFA$="←"THENPOKE36879,25:GOTO80
820 PRINT:VERIFY"",1,1:PRINT"TASTE":G
OSUB61000
830 POKE36879,25:GOTO80
840 POKE36879,24:PRINT"BESTAETIGEN":
GOSUB61000:IFA$="←"THENPOKE36879,25:GOTO
80
845 PRINT"!! INITIALISIERE !!"
850 FORA=0TO9:PRINT"SPRITE"A:FORB=0TO31
:POKENZ+1024+A*104+B,0:NEXTB
860 POKE4608+A*22+20,0
870 FORB=0TO17:POKE4608+A*22+B,32:NEXTB,
A:POKE36879,25:GOTO80
880 INPUT"SPRITENUMMER:";S%:PRINT"RIC
HTIG?":GOSUB61000
890 IFA$="N"THEN880
```

Listing 2. Der Sprite-Generator



```
891 IFA$="←"THEN80
900 PRINT"SPRITE-DEMO!"
910 POKE36869,205
920 FORA=-127TO127:POKE673,ABS(A):POKE674,ABS(A):POKE683,S%:SYS8021
930 NEXTA:POKE683,S%:SYS8099:POKE36869,192:GOTO80
940 PRINT"WIRKLICH?":GOSUB61000
950 IFA$="J"THENNEW
960 GOTO80
999 END
40000 PRINT"!! WERTE AUS !!";
40005 FORA=0TO15:BB%=0:FORB=0TO7
40010 BB%=BB%-((PEEK(B%+A*22+B)AND127)=81)*2↑(7-B):NEXTB
40020 POKENZ+128*8+S%*104+A,BB%:NEXTA
40030 B%=B%+8
40040 FORA=0TO15:BB%=0:FORB=0TO7
40050 BB%=BB%-((PEEK(B%+A*22+B)AND127)=81)*2↑(7-B):NEXTB
40060 POKENZ+128*8+S%*104++16+A,BB%:NEXTA
40070 RETURN
50000 PRINT"SPRITENR."S%
50010 PRINT"┌────────────────┐"
50020 FORA=0TO15:PRINT"│                │":NEXT:PRINT"└────────────────┘"
50030 B%=4096+22*3+3:X%=0:Y%=0
50035 D%=B%+Y%*22+X%
50040 IF(PEEK(D%)AND128)=0THENPOKED%,PEEK(D%)OR128:POKED%+33792,6:GOTO50060
50050 IF(PEEK(D%)AND128)THENPOKED%,PEEK(D%)AND127:POKED%+33792,6
50060 AX%=X%:AY%=Y%:GOSUB61000
50070 IFA$=""ANDY%<15THENY%=Y%+1:GOTO50200
50080 IFA$=""ANDY%>0THENY%=Y%-1:GOTO50200
50090 IFA$=""ANDX%<15THENX%=X%+1:GOTO50200
50100 IFA$=""ANDX%>0THENX%=X%-1:GOTO50200
50110 IFA$=" "THENPOKED%,209:POKED%+33792,6:GOTO50060
50120 IFA$=CHR$(20)THENPOKED%,160:POKED%+33792,6:GOTO50060
50130 IFA$=""THENPRINT"";:FORA=0TO15:PRINT"                ":NEXT:GOTO50030
50140 IFA$="I"THENGOSUB60000:GOTO50035
50150 IFA$=CHR$(13)THEN50240
50160 IFA$="←"THENRETURN
50200 D%=B%+AY%*22+AX%
50210 IF(PEEK(D%)AND128)=0THENPOKED%,PEEK(D%)OR128:GOTO50230
50220 IF(PEEK(D%)AND128)THENPOKED%,PEEK(D%)AND127
50230 GOTO50035
50240 PRINT"WIRKLICH(J/N)?";
50250 GOSUB61000:IFA$="J"THENRETURN
50260 PRINT"";:GOTO50000
60000 FORX=0TO15:FORY=0TO15:DD%=B%+22*Y+X
60010 IF(PEEK(DD%)AND127)=81THENPOKEDD%,32:POKEDD%+33792,6:NEXT:NEXT:RETURN
60020 IF(PEEK(DD%)AND127)=32THENPOKEDD%,81:POKEDD%+33792,6:NEXT:NEXT:RETURN
61000 POKE198,0:WAIT198,1:GETA$:RETURN
READY.
```

Listing 2. Der Sprite-Generator (Schluß)



Hex-DATA-Automat

Der Computer programmiert sich selbst — Maschinenprogramme werden automatisch in DATA-Statements mit Prüfsumme umgewandelt.

Ein Maschinenprogramm in ein korrektes Basic-Ladeprogramm umzusetzen, ist sicherlich eine sehr langweilige Programmieraufgabe, außerdem schleichen sich sehr schnell Fehler ein.

Soll diese Umsetzung automatisch erfolgen, müßte sich der Computer — salopp gesagt — selbst programmieren. Dies ist prinzipiell möglich; doch zuvor einige Grundlagen.

Geben Sie hierzu das nebenstehende kleine Testprogramm ein.

```
100 POKE 2,0
110 ZL=PEEK(2) : POKE 2,ZL+1
120 D$=STR$(ZL+1000)
130 D$=D$+"DATA ABCFDEF"
140 PRINT CHR$(147);D$
150 PRINT "RUN 110"
160 :
170 :
180 END
```

```
1 REM ****************************************
2 REM *           ++++ PROGRAMM-NAME ++++           *
3 REM *                                              *
4 REM *   STICHWORT..........................         *
5 REM *   COPYRIGHT (C) ....................         *
6 REM *                                              *
7 REM ****************************************
8 :
9 :
10 DIM H(75) : FOR I=0 TO 9
20 H(48+I)=I : H(65+I)=I+10 : NEXT
30 FOR I=ANFANG TO SCHLUSS: READ A$ : REM HIER AKTUELLE WERTE EINSETZEN !
40 H=ASC(LEFT$(A$,1)):L=ASC(RIGHT$(A$,1))
50 D=H(H)*16+H(L) : S=S+D : POKE I,D
60 A=A+1:IF A<20 THEN NEXT : A=-1
65 PRINT "ZEILE:";1000+Z;
70 READ V : Z=Z+1 : IF V=S THEN 85
80 PRINT" PRUEFSUMMENFEHLER !";999+Z:STOP
85 IF A<0 THEN END
90 S=0 : A=0 : PRINT : NEXT : END
95 :
96 :
97 :
98 :
99 :
200 REM ****************************************
210 REM *  HEX-DATA-AUTOMATIK (VC 20 & C 64)  *
220 REM *                                          *
230 REM *  DATA-ZEILEN PROGRAMMIEREN              *
240 REM *       START MIT RUN 200                 *
250 REM *                                          *
260 REM ****************************************
270 :
280 :
500 INPUT "START-ADRESSE";A
510 H=INT(A/256):L=A-H*256:POKE 640,H:POKE 641,L
520 INPUT "END-ADRESSE    ";E
530 H=INT(E/256):L=E-H*256:POKE 642,H:POKE 643,L
540 POKE 2,1 : IF A>E THEN 500
550 :
560 A=PEEK(640)*256+PEEK(641)
570 E=PEEK(642)*256+PEEK(643)
580 :
590 DIM H$(20):FOR I=0 TO 9 : H$(I)=CHR$(I+48)
600 H$(I+10)=CHR$(I+65)     : NEXT : Z=3
610 :
620 FOR I=1 TO 20 : D=PEEK(A) : S=S+D
630 H=INT(D/16)      : L=D-16*H
640 A$=A$+H$(H)+H$(L)+","
650 IF A=E THEN Z=2 : D=PEEK(2) : GOTO 670
660 A=A+1 : NEXT : D=PEEK(2) : POKE 2,D+1
670 A$=STR$(999+D)+" DATA "+A$
680 A$=A$+STR$(S)
690 H=INT(A/256):L=A-H*256:POKE 640,H:POKE 641,L
700 POKE 631,19: POKE 632,13 : POKE 633,13
710 POKE 198,Z : PRINT CHR$(147);A$
720 PRINT "RUN 560" : END

READY.
```

Listing »Hex-DATA-Automat«

Die Programmzeile 100 setzt die Speicheradresse 2 auf Null. Anschließend wird der Wert dieser Adresse nach ZL geholt und die Adresse um eins erhöht. Der STR$-Befehl wandelt den Wert ZL+1000 in einen String, und die Zeile 130 erweitert den String mit »DATA ABCDEF«. Die CHR$-Anweisung löscht anschließend den Bildschirm und schreibt den String »1000 DATA ABCDEF« links oben auf den Bildschirm. Zuletzt wird in der zweiten Bildschirmzeile der Text »RUN 110« gedruckt.

Falls Sie nach RUN die Taste HOME drücken, steht der Cursor auf der Zeile »1000 DATA ABCDEF«. Drücken Sie nun die RETURN-Taste, dann wird die Zeile 1000 in das Programm aufgenommen. Der Cursor steht jetzt auf dem »RUN 110«. Drücken Sie jetzt erneut RETURN, so startet das Programm wieder, und es folgt der nächste Durchgang mit:
1001 DATA ABCDEF
RUN 110

Da im ersten Durchlauf der Wert in der Speicherzelle 2 um eins erhöht wurde, lautet die nächste Zeilennummer 1001. Nun könnten Sie wieder (in Handarbeit) HOME/RETURN/RETURN eingeben, doch — und jetzt wird's interessant — auch dies kann der Computer durchführen.

160 POKE 198,3
170 POKE 631,19:POKE 632,13:POKE 633,13

Geben Sie nun RUN ein. Das Programm erweitert sich nun automatisch — ab der Nummer 1000 — um DATA-Zeilen.

Dies ist möglich, da alle Commodore-Computer mit einem Tastaturpuffer arbeiten. In diesem Zwischenspeicher, der beim VC 20 und C 64 ab der Adresse 631 beginnt, kann sich der Computer bis zu neun Tastatureingaben merken. Die Anzahl der Zeichen in dem Puffer steht in der Adresse 198.

In der vorherigen Programmerweiterung wurde in der Zeile 160 der Wert 3 eingePOKEt. Der Computer meint anschließend, es seien drei Tastatureingaben erfolgt. Die POKE-Befehle in der Zeile 170 simulieren die Eingabesequenz HOME, RETURN, RETURN. Nach dem Programmende vergißt der Computer diese untergeschobenen Eingaben keineswegs, sondern führt sie nachträglich aus.

Aus der Zeit des legendären PET 2001 stammt noch die Bezeichnung »selbsterhaltendes Programm«. Na ja, aber so hatte das Kind wenigstens einen Namen.

Leider hat das beschriebene Verfahren den Nachteil, daß der Computer, sobald er sich selbst die Zeile einprogrammiert, die Variablen löscht. Aus diesem Grund müssen Sie wichtige Werte vor dem Programmabbruch durch POKE sichern und beim Neustart mit PEEK zurückholen. In dem vorherigen Testprogramm wurde beispielsweise der Zähler für die Zeilennummer mit der Adresse 2 weiter gegeben.

Das Programm »HEX-DATA-Automatik« arbeitet im Prinzip genau nach dem zuvor beschriebenen Verfahren. Die Umwandlungsroutine wird mit RUN 200 gestartet. Das Programm fragt dann nach der Anfangs- und Endadresse des Maschinenprogramms und wandelt es anschließend in DATA-Zeilen um. Diese haben das folgende Format:

1000 DATA 01,02,03,04,05,... ,20, 1234
1001 DATA 11,12,13,14,15,... ,40, 5678
1002 DATA ...

Nach jeweils 20 Hexadezimal-Daten folgt immer eine Prüfsumme.

Anschließend müssen Sie die DATA-Routine (200—720) löschen und den Schleifenzähler in Zeile 30 anpassen. Falls das Programm veröffentlicht werden soll, können Sie zusätzlich ein Copyrightstatement hinzufügen.

Für diesen Zweck wurde diese Routine auch ursprünglich erstellt. Die Zeilen 10—90 wandeln die Hex-Zahlen wieder um und schreiben das Maschinenprogramm in den entsprechenden Speicherbereich zurück (Zeile 30 beachten).

Das Programm kann in dieser Form sehr leicht abgetippt werden, da Prüfsummenfehler wie folgt angezeigt werden:

ZEILE 1000
ZEILE 1001
ZEILE 1002
ZEILE 1003 PRÜFSUMMENFEHLER !
BREAK IN 80

Anschließend müssen nur 20 Daten überprüft werden, so daß auch längere Basic-Lader vergleichsweise schnell und fehlerfrei abgeschrieben werden können.

(Heino Velder/ev)

Dem »Springvogel« auf die Sprünge geholfen

```
59300 A=0:I=827
59310 READ N:IF N<0 THEN 59340
59320 A=A+N:I=I+1
59330 POKE I,N:GOTO 59310
59340 IF I<>953 THEN PRINT"FALSCHE DATA-ZAHL IN 59400 FF. DIFF.:";I-953:STOP
59350 IF A<>15594 THEN PRINT"DATA-ERROR IN 59400 FF. DIFF.:";A-15594:STOP
59360 SYS 828
59400 DATA 120,169, 75,141, 20,  3,169
59410 DATA   3,141, 21,  3, 88, 96, 10
59420 DATA  70,169, 53,133,  1,206, 73
59430 DATA   3,208, 32,169,  2,141, 73
59440 DATA   3,162,  8,189, 71,226, 24
59450 DATA 106,144,  2,  9,128,157, 71
59460 DATA 226,189, 79,226, 24, 42,105
59470 DATA   0,157, 79,226,202,208,231
59480 DATA 172, 88,226,162,  0,189, 89
59490 DATA 226,157, 88,226,232,224,  7
59500 DATA 208,245,140, 95,226,172,103
59510 DATA 226,189, 95,226,157, 96,226
59520 DATA 202,208,247,140, 96,226,206
59530 DATA  74,  3,208, 24,173,137,226
59540 DATA 208,  6,162, 70,169,126,208
59550 DATA   4,162, 15,169,  0,141,137
59560 DATA 226,141,142,226,142, 74,  3
59570 DATA 169, 55,133,  1, 76, 49,234
59580 DATA  -1

READY.
```

Der »Springvogel« sieht sich ja zahlreichen Unannehmlichkeiten ausgesetzt, die recht aktiv versuchen, seinen Sprüngen ein Ende zu setzen. Aber Bänder und Aufzüge transportieren ihn, ohne daß sie sich selbst bewegen, und warum sind eigentlich die Fallen so tödlich?

Wer hier Alternativen haben möchte, braucht nur das Programm um die nebenstehenden Zeilen zu erweitern: Einfach »Springvogel« laden, Ergänzung dazutippen, abspeichern (!), starten.

Die zusätzliche Bewegung auf dem Bildschirm dürfte den Springvogel noch etwas attraktiver machen: Bänder und Aufzüge bewegen sich nun tatsächlich, und die Fallen zwinkern zumindest diskret. Erreicht wird das durch ein kleines Maschinensprachprogramm im Kassettenpuffer, das im Interrupt mitläuft und systematisch die betreffenden Zeichen im neuen Zeichensatz ändert. Die Geschwindigkeit von Bändern und Aufzügen wurde, soweit möglich, der des Vogels angepaßt. Wen dabei das Rasen der Aufzüge nervös macht, der braucht nur den dritten DATA-Wert in Zeile 59430 (32) in 66 und den Kontrollwert in Zeile 59350 (15594) entsprechend in 15628 zu ändern.

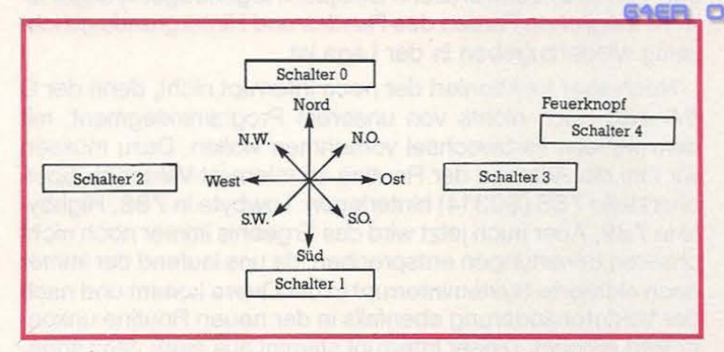
(Thomas Schmidt/aa)

Joystick-Abfrage in Theorie und Praxis

Wenn der Joystickanschluß für Sie »ein Buch mit sieben Siegeln« ist, so wird Ihnen dieser Artikel für den VC 20 weiterhelfen. Doch auch die »Profis« werden einige wichtige Informationen finden.

Ein Joystick besteht aus vier Schaltern, die im rechten Winkel zueinander angeordnet sind. Der Handgriff erlaubt neun abfragbare Positionen:
— eine Position mit allen Schaltern offen: Griff in Ruhe
— vier Positionen mit je einem Schalter geschlossen: Griff in Nord, Süd, Ost, West
— vier Positionen mit zwei Schaltern geschlossen: Griff in Nordost, Südost, Südwest, Nordwest

Ein zusätzlicher »Feuerknopf« hat einen eigenen Schalter. Grafisch sieht das so aus:



Jeder der fünf Schalter ist mit je einer Leitung von zwei speziellen integrierten Bausteinen mit dem Namen »VIA 6522« (Versatile Interface Adapter) verbunden. Diese sind, wie der Name andeutet, programmierbare Adapter für die Ein- und Ausgabe (also auch für den Joystick).

Leider sind die fünf Schalter etwas ungleichmäßig auf die beiden VIAs verteilt:
— Schalter 0, 1, 2 und der Feuer-Schalter 4 verwenden VIA 1
— Schalter 3 verwendet VIA 2

Der Kontakt des Joysticks mit den VIAs und damit mit dem VC 20 wird durch zum Teil Ihnen schon bekannte Registerzellen geregelt, welche folgende Adressen haben:
— Ein-/Ausgabe-Register A des VIA 1: 37137
— Ein-/Ausgabe-Register B des VIA 2: 37152

Der Vollständigkeit halber sei erwähnt, daß jeder VIA noch ein zweites E/A-Register hat, nämlich:
— E/A-Register B des VIA 1: 37136
— E/A-Register A des VIA 2: 37153

Für den Joystick brauchen wir diese jedoch nicht.

Die einzelnen Leitungsanschlüsse der Joystick-Schalter sind:
— Schalter 0 ... Bit 2 von 37137
— Schalter 1 ... Bit 3 von 37137
— Schalter 2 ... Bit 4 von 37137
— Schalter 4 ... Bit 5 von 37137
— Schalter 3 ... Bit 7 von 37152

Ich habe gesagt, daß 37137 und 37152 Ein- und Ausgaberegister sind, das heißt wir können sie in beiden Richtungen benützen.

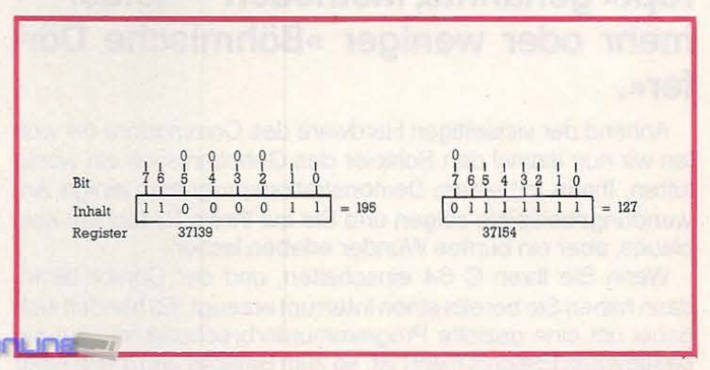
Die Entscheidung darüber liegt in je einem zugeordneten »Daten-Richtungs-Register«.
— Dem E/A-Register 37137 ist das DR-Register 37139 zugeordnet.
— Dem E/A-Register 37152 ist das DR-Register 37154 zugeordnet.

Dieses Arrangement erlaubt, jede einzelne Leitung eines E/A-Registers separat auf Ein- oder Ausgabe zu schalten, völlig unabhängig voneinander.

Das geht so:

Sobald in einem Bit des DR-Registers eine 1 steht, ist die entsprechende Leitung des E/A-Registers auf Ausgabe geschaltet, bei einer 0 auf Eingabe.

Im Bild unten habe ich die notwendigen Bitmuster in die beiden DR-Register eingezeichnet.



Im Register 37139, an dem ja vier Schalter hängen, wäre die hineinzuPOKEnde Zahl 195. Da aber während der Joystick-Abfrage dieses Register für nichts anderes verwendet wird, setzen wir ruhig das ganze Register auf 0.

```
10 POKE 37139,0
```

Beim Register 37154 ist die Lage anders, da das zugehörige E/A-Register 37152 zur Tastaturabfrage verwendet wird. Da müssen wir die Auswahl der Leitung schon genau machen.

```
20 POKE 37154,127
```

Während der Joystick-Abfrage funktionieren die Tasten in der Spalte 127 der 8 x 8-Matrix nicht.

Zeile 10 und 20 »initialisieren« die Joystick-Abfrage.

Ein Experimentier-Programm zum Testen, was jetzt bei der Bewegung des Joysticks stattfindet, ist ganz einfach:

```
30 PRINT PEEK(37137);PEEK(37152)
40 GOTO 30
```

Am Ende empfiehlt es sich, die volle Tastatur wieder einzuschalten mit

```
POKE 37154,255
```

In einem Programm können Sie also einfach die Werte in den beiden Adressen der Zeile 30 (mit IF .... THEN) abfragen.

Zur Vermeidung von eventuellen Störungen durch die Mehrfachfunktionen der Register ist es aber empfehlenswert, die einzelnen Bits direkt abzufragen. Das sieht dann so aus:

```
10 POKE 37139,0
20 POKE 37154,127
30 IF PEEK(37137)   AND    4 THEN.....   (Schalter 0)
40 IF PEEK(37137)   AND    8 THEN.....   (Schalter 1)
50 IF PEEK(37137)   AND   16 THEN.....   (Schalter 2)
60 IF PEEK(37137)   AND   32 THEN.....   (Schalter 4 = Feuer-
                                          knopf)
70 IF PEEK(37152)   AND  128 THEN.....   (Schalter 3)
80 POKE 37154,255
```

(Helmuth Hauck/rg)



Unterbrechen Sie mich bitte!

Im Gegensatz zum Bereich zwischenmenschlicher Beziehungen, wo jemanden zu unterbrechen als plumpe Unhöflichkeit eingestuft wird, ist dies einem Computer gegenüber nicht nur ein beliebtes, sondern sogar erwünschtes Verfahren effektvoller Programmgestaltung. Für die meisten Anwender aber sind solche, »Interrupt« genannte, Methoden — leider — mehr oder weniger »Böhmische Dörfer«.

Anhand der vielseitigen Hardware des Commodore 64 wollen wir nun einmal den Schleier des Geheimnisses ein wenig lüften, Ihnen mit einem Demonstrationsprogramm einige Anwendungsbeispiele zeigen und Sie auf Ihrem Bildschirm kein blaues, aber ein buntes Wunder erleben lassen.

Wenn Sie Ihren C 64 einschalten, und der Cursor blinkt, dann haben Sie bereits einen Interrupt erzeugt. Es handelt sich dabei um eine gezielte Programmunterbrechung, die auf ein bestimmtes Ereignis fixiert ist, so zum Beispiel den Inhalt einer Speicherstelle. Tritt der erwünschte Zustand ein, so räumt die Hardware diesem Vorgang Priorität vor allen anderen Aufgaben ein. Gleich was der Computer im Augenblick macht, er wird unverzüglich seine Tätigkeit einstellen und ein spezielles Unterprogramm abarbeiten, das ihm zuvor als Interruptroutine deklariert wurde. Im Betriebssystem-ROM befindet sie sich von Adresse $EA31 (dezimal 59953) bis $ECB8 (60600). Hier wird etwa 60mal in jeder Sekunde geprüft, ob eine — und falls ja, welche — Taste gedrückt wurde; hier wird das Blinken des Cursors erzeugt und der Motor der Datasette ein- oder ausgeschaltet. Wenn der Interrupt beendet ist, setzt der Computer die Bearbeitung des laufenden Programms exakt an der Stelle fort, an der er sich vor Eintritt des Interrupts gerade befand. Gewöhnlich merkt man von solchen Intermezzi nichts, weil der Prozessor eine ungeheure Geschwindigkeit entwickelt, deren Arbeitstakte sich bereits im Bereich von Mikrosekunden (millionstel Sekunden) bewegen.

Noch eine andere Einrichtung der von Ihnen benutzten Geräte ist mit extremer Schnelligkeit ausgestattet: Der Elektronenstrahl, der vor Ihren Augen 25mal in jeder Sekunde ein neues Bild auf die Mattscheibe zeichnet. Nach diesen grundsätzlichen Überlegungen wollen wir aus solchen Voraussetzungen einen Wettstreit der Systeme entwickeln und den Computer gegen den flotten Strahl antreten lassen. Unglaublich, werden Sie jetzt wahrscheinlich sagen, aber warten Sie ab! Möglich ist das nämlich, weil der für die Bildorganisation zuständige Video-Interface-Controller (VIC) stets genau darüber »im Bild« ist, welche Rasterzeile des Monitorbildes augenblicklich geschrieben wird. Das ist die Bedingung dafür, daß auf dem Schirm auch ein Bild im geordneten Zusammenhang wiedergegeben wird. Der VIC, dessen Register unter den Adressen 53248 ($D000) bis 53294 ($D02E) erreichbar sind, führt in der Speicherzelle 53266 ($D012) Buch über die jeweilige Zeilennummer. Darüber war schon einmal zu lesen, daß es unsinnig wäre, dort etwas hineinzuschreiben, weil man den Elektronenstrahl der Bildröhre gar nicht steuern könne. Und das ist nur zum Teil richtig, denn steuern können wir den Strahl nicht, wohl aber können wir ihn steuern lassen.

Dafür ist besagtes Register — wie übrigens viele andere auch — mit einer Doppelfunktion ausgerüstet. Abhängig von der jeweiligen Zugriffsart, Lesen oder Schreiben (PEEK oder POKE), erreicht man unter derselben Hausnummer verschiedene Adressaten. Wird das Register gelesen, so erfährt man die gerade bearbeitete Rasterzeile, wird es beschrieben, bleibt der übermittelte Wert gespeichert und dient dem internen Vergleich, ob er mit der aktuellen Zeile identisch ist. Wenn dieser Fall eintritt, so reagiert die Hardware selbständig darauf und erzeugt einen Interrupt — falls ein solcher vorgesehen war. Zuvor muß dem Computer nämlich noch mitgeteilt werden, daß dies ein Interruptauslöser sein soll. Die Anmeldestelle ist die Adresse 53274 ($D01A), das Interrupt-Masken-Register.

Diese Speicherstelle korrespondiert ständig mit ihrem Nachbarn 53273 ($D019), dem Interrupt-Request-Register. Ist die vorgewählte Rasterzeile erreicht, signalisiert 53266 dieses Ereignis durch Kippen des Bits 0 im Request-Register: Es nimmt den Wert 1 an. Ist Bit 0 auch in der Maske gesetzt, wird der Interrupt-Pin des VIC aktiv und löst die Unterbrechung aus. Im Demo-Programm schreiben wir zu diesem Zweck eine 1 in die Maske, und fortan gilt für uns die Gesetzmäßigkeit, daß Bildhintergrund und -rand unifarben dargestellt werden, nicht mehr. Wir ändern nämlich ab einer beliebigen Rasterzeile die Bildfarben, um sie einige Zeilen weiter abermals umzuschalten. Da das schneller vor sich geht, als das träge menschliche Auge es registrieren kann, resultiert daraus kein wirres Flackern, sondern ein konstantes mehrfarbiges Bild, das im Extremfall (siehe Beispiel »Regenbogen«) sogar alle 16 möglichen Farben des Randes und Hintergrunds gleichzeitig wiederzugeben in der Lage ist.

Noch aber funktioniert der neue Interrupt nicht, denn der C 64 weiß noch nichts von unserem Programmsegment, mit dem wir den Farbwechsel vornehmen wollen. Dazu müssen wir ihm die Adresse der Routine im Interrupt-Vektor ab Speicherstelle 788 ($0314) hinterlegen: Lowbyte in 788, Highbyte in 789. Aber auch jetzt wird das Ergebnis immer noch nicht unseren Erwartungen entsprechen, da uns laufend der immer noch aktivierte Systeminterrupt in die Quere kommt und nach der Vektorenänderung ebenfalls in der neuen Routine unkontrolliert arbeitet. Dieser Interrupt stammt aus einer ganz anderen Quelle, vom Complex-Interface-Adapter (CIA), auf den wir gleich noch zu sprechen kommen. Softwaremäßig können wir ihn durch Setzen der Interruptflagge (SEI) nicht unterbinden, weil damit auch der ebenfalls maskierbare Rasterzeilen-Interrupt abgeschaltet würde. Zwei Möglichkeiten gibt es, das Problem zu lösen: Der CIA-Interrupt wird belassen, muß dann aber zu Beginn der Interrupt-Routine abgefragt (zum Beispiel durch Prüfen des Bits 0 im Request-Register 53273) und gegebenenfalls durch eine Umleitung über Sprungbefehle unschädlich gemacht werden, oder er wird eliminiert durch Schreiben des Wertes 127 in die Speicherstelle 56333 ($DCOD). Weil wir in unserem Demo-Programm weder Cursor-Blinken noch die Tastaturabfrage benötigen, entscheiden wir uns für letztere und löschen gleich zu Programmbeginn (siehe Assembler-Listing) den CIA-Interrupt völlig.

Außerdem ist zu berücksichtigen, daß der Elektronenstrahl auf allen Seiten ein Stück über den Rand des auf dem Monitor sichtbaren Bildes hinausschreibt. Dadurch entstehen Zeilennummern bis 280, so daß von Register 53266 aus bei einem Überlauf ein Highbyte nach Bit 7 des Registers 53265 übertragen werden muß. In dieser als »SCREEN« definierten Speicherstelle liegen allerdings auch einige andere Funktionen, die gegebenenfalls durch eine logische OR-Verknüpfung be-

Basic-Lader von »Rasterzeilen-Interrupt«

```
5 RUN100
10 ***********************************
15 *                                 *
20 *    RASTERZEILEN-INTERRUPT       *
25 *                                 *
30 *       (BASIC-LOADER)            *
35 *                                 *
40 *     EIN   DEMO-PROGRAMM         *
45 *                                 *
50 *   FUER DEN  COMMODORE 64        *
55 *                                 *
60 *       VON HELMUT WELKE          *
65 *                                 *
70 *      MOEHNESTRASSE 26           *
75 *                                 *
80 *      5760  ARNSBERG 1           *
85 *                                 *
90 *      TEL. (02932) 23627         *
95 *                                 *
99 ***********************************
100 :
110 PRINTCHR$(147)"LOADING":AD=49152
120 READX:IFX<0GOTO300
130 POKEAD,X:AD=AD+1:S=S+X:GOTO120
200 :
300 IFAD=49737ANDS=64373THENSYS49152
500 PRINT:PRINT"DATAS FEHLERHAFT"
600 :
700 :
800 :
1000 DATA 32,68,229,169,127,141,13,220
1010 DATA 141,4,220,141,5,220,162,0
1020 DATA 142,26,208,202,134,251,169
1030 DATA 11,141,17,208,133,253,169
1040 DATA 14,141,33,208,133,254,169
1050 DATA 44,160,193,141,20,3,140,21
1060 DATA 3,160,1,140,18,208,140,26
1070 DATA 208,136,169,25,32,21,193,166
1080 DATA 254,169,2,32,21,193,232,134
1090 DATA 254,224,200,144,244,166,253
1095 :
1100 DATA 169,2,32,21,193,232,134,253
1110 DATA 224,115,144,244,162,7,160
1120 DATA 13,24,32,240,255,169,200,160
1130 DATA 193,32,30,171,169,150,160
1140 DATA 0,32,21,193,169,27,141,17
1150 DATA 208,169,200,32,21,193,140
1160 DATA 26,208,140,32,208,32,68,229
1170 DATA 162,4,142,33,208,160,3,24
1180 DATA 32,240,255,169,254,160,193
1190 DATA 32,30,171,169,30,160,0,32
1195 :
1200 DATA 21,193,169,88,160,193,141
1210 DATA 20,3,140,21,3,162,25,134,253
1220 DATA 142,18,208,173,25,208,141
1230 DATA 25,208,160,1,140,26,208,136
1240 DATA 169,2,32,21,193,232,134,253
1250 DATA 224,170,144,244,169,100,32
1260 DATA 21,193,32,68,229,140,26,208
1270 DATA 169,115,160,193,141,20,3,140
1280 DATA 21,3,173,25,208,141,25,208
1290 DATA 160,0,132,251,200,140,18,208
1295 :
1300 DATA 140,26,208,169,0,32,21,193
1310 DATA 169,15,141,134,2,162,3,24
1320 DATA 32,240,255,169,221,160,193
1330 DATA 32,30,171,169,20,160,0,32
1340 DATA 21,193,206,134,2,16,237,169
1350 DATA 1,168,32,21,193,76,0,192,141
1360 DATA 6,220,140,7,220,169,17,141
1370 DATA 14,220,169,89,141,15,220,173
1380 DATA 15,220,74,176,250,96,165,251
1390 DATA 240,9,16,15,169,0,170,164
1395 :
1400 DATA 253,208,14,169,2,162,1,164
1410 DATA 254,208,6,169,7,162,255,160
1420 DATA 1,134,251,140,18,208,141,32
1430 DATA 208,173,25,208,141,25,208
1440 DATA 76,129,234,173,33,208,73,2
1450 DATA 168,41,2,240,6,165,253,24
1460 DATA 105,21,44,165,253,141,18,208
1470 DATA 140,33,208,76,79,193,198,251
1480 DATA 166,251,16,4,162,15,134,251
1490 DATA 189,152,193,72,188,184,193
1495 :
1500 DATA 189,168,193,170,104,142,18
1510 DATA 208,140,17,208,141,32,208
1520 DATA 141,33,208,76,79,193,4,11
1530 DATA 5,9,3,10,1,14,7,15,2,13,0
1540 DATA 8,6,12,1,11,254,235,219,203
1550 DATA 179,162,138,122,106,82,75
1560 DATA 59,51,33,27,155,27,27,27,27
1570 DATA 27,27,27,27,27,27,27,27,27
1580 DATA 27,14,8,31,196,69,77,79,45
1590 DATA 208,82,79,71,82,65,77,77,17
1595 :
1600 DATA 17,17,17,17,13,29,29,29,29
1610 DATA 29,29,29,29,29,210,193,211
1620 DATA 212,197,210,218,197,201,204
1630 DATA 197,206,45,201,78,84,69,82
1640 DATA 82,85,80,84,0,156,213,78,68
1650 DATA 32,78,85,78,32,32,32,32,17
1660 DATA 17,17,83,69,72,69,78,32,211
1670 DATA 73,69,32,32,32,32,17,17,17
1680 DATA 78,79,67,72,32,68,69,78,13
1690 DATA 17,17,17,17,17,29,29,29,29
1695 :
1700 DATA 29,29,29,29,29,29,210,160
1710 DATA 197,160,199,160,197,160,206
1720 DATA 160,194,160,207,160,199,160
1730 DATA 197,160,206,0
1740 DATA -1

READY.
```

rücksichtigt werden müssen. Das Demo-Programm verwendet diese Funktionen indem durch Löschen des Bits 4 der Bildschirm ausgeblendet wird, so daß auf dem Schirm nur noch ein ganzflächiger Rand erscheint, obwohl auch weiterhin Texte auf den jetzt unsichtbaren Hintergrund ausgegeben werden können. Beispiel: POKE 53265, PEEK (53265) AND 255-16 läßt den Hintergrund verschwinden, POKE 53265, PEEK(53265) OR 16 zaubert ihn dann wieder herbei.

Der Rasterzeilen-Interrupt ist eine Spezialität des C 64, die ihn zu einem äußerst vielseitigen Gerät macht. Die Programmier-Profis der Videospiel-Produzenten benutzen ihn nicht allein dafür, oben blauen Himmel und unten braune Erde darzustellen, sondern auch, um ein nur partielles Scrolling zu erzielen, um Text und hochauflösende Grafik zu mischen, um gleichzeitig normale und Multicolorzeichen zu benutzen und und und... Aber damit erschöpfen sich die Interruptmöglichkeiten des Commodore 64 noch nicht. Bit 1 (Wert = 2) der Mas-

ke in 53274 erzeugt einen Interrupt, wenn ein Sprite Berührung mit einem Zeichen hat, Bit 2 (Wert = 4) wenn Sprite mit Sprite zusammenstößt, Bit 3 (Wert = 8) wenn ein Impuls vom Lightpen kommt, oder der Feuerknopf eines am Controlport 1 angeschlossenen Joysticks gedrückt wird und schließlich Bit 7 (Wert = 128), wenn eines der genannten Ereignisse eingetreten ist.

Und dann ist da noch der bereits genannte CIA-Interrupt, der von einem gleich doppelt vorhandenen Baustein stammt. Beiden sind jedoch im C 64 teilweise unterschiedliche Aufgaben zugewiesen. CIA 1 hat die Basisadresse 56320 ($DC00), CIA 2 eine solche von 56576 ($DD00). Hier erfolgt beispielsweise die Tastaturdekodierung, die Abfrage von Joysticks, Paddles und Lightpen, die serielle Datenübertragung zu einer Schnittstelle, hier befinden sich die Echtzeituhren und die Timer. Wenden wir uns letzteren in CIA 1 zu.

Der Timer ist ein 16-Bit-Zählregister, das nach dem Starten ohne weiteres Zutun mit einer konstanten Geschwindigkeit dekrementiert, das heißt jeweils um 1 abwärts gezählt wird. Durch die zuvor beim VIC schon erwähnte Doppelfunktion besteht die Möglichkeit, den Timer ab einem bestimmten Wert zählen zu lassen: Lesen der Adresse 56324 ($DC04) liefert den aktuellen Stand des Lowbytes, Hineinschreiben den Startwert des Timers, ebenso beim Highbyte unter der Adresse 56325. Auf diese Weise wird der Systeminterrupt erzeugt, da der Computer in der Initialisierungsphase den Timer mit dem Wert 16421 (= 37 low und 64 high) lädt. Wenn der Timer über Null hinauszählt und damit einen sogenannten Unterlauf erzeugt, wird das durch Setzen des Bits 0 im Interrupt-Control-Register 56333 ($DC0D) signalisiert. Auch dieses arbeitet wieder doppelt: Lesen ergibt die Interruptanforderung, ein Schreibzugriff erzeugt die Maske, die darüber entscheidet, welches Ereignis Interruptauslöser sein soll. Besondere Beachtung beim Beschreiben des Registers verdient Bit 7, das darüber bestimmt, ob die nachfolgend gesetzten Bits 6 bis 0 in der Maske gesetzt oder gelöscht werden. Alle übrigen bleiben unangetastet. Deshalb löscht 127 (Bit 7 nicht gesetzt!) im Demo-Programm sämtliche Maskenbits (siehe Bitmuster im Assembler-Listing, Zeile 1010), deshalb setzt 129 (Bitmuster 10000001) das Bit 0 der Maske und schaltet damit auf Interrupt durch den Timer. Erzeugen Sie doch einmal im Direktmodus auf dem Bildschirm ein längeres Zählintervall durch Heraufsetzen des Timer-Highbytes: POKE 56325, 255 läßt den Cursor sehr träge werden, POKE 56325, 5 versetzt ihn in nervöses Flattern.

Dieser Timer besitzt einen Zwillingsbruder mit den Adressen 56326 und 56327, der nicht nur gleichartig konstruiert ist und ebenfalls eigenständig einen Interrupt auf Bit 1 des Control-Registers erzeugen kann, sondern sich auch mit dem Timer A koppeln läßt. Beide Timer können nämlich auf verschiedene Taktquellen gelegt, unterschiedlich getriggert werden. Im Demo-Programm nutzen wir das aus, indem wir Timer A Systemtakte zählen lassen (Bit 5 des Registers 56334 gelöscht — Standardeinstellung), Timer B hingegen nur die Unterläufe von Timer A durch Setzen des Bits 6 im Register 56335. Dadurch erhalten wir einen 32-Bit-Zähler, der beliebige Zeitverzögerungen ermöglicht, eleganter als mit jeder ausschließlich softwaremäßig realisierten Warteschleife, weil die Timer von keinem Interrupt unterbrochen werden.

In der Warteschleife des Demo-Programms (Listing ab Zeile 9010), die als Subroutine angelegt ist, wird mit den Timern kein Interrupt erzeugt, sondern lediglich eine Zeitverzögerung erzielt. Dazu wird zunächst Timer B mit den vom Hauptprogramm im Akku und im Y-Register übergebenen Werten geladen, während Timer A konstant mit einem mittleren Wert arbeitet. Dann werden beide gestartet durch Setzen des Bits 0 im Register 56334 für Timer A und 56335 für Timer B. Aus Zeile 9030 des Listings ist ersichtlich, daß Bit 3 für Timer A gelöscht ist, was Continuous- oder Dauerbetrieb zur Folge hat. Jedesmal, wenn der Timer einen Unterlauf hat, lädt er umgehend wieder den zwischengespeicherten Startwert und beginnt erneut zu zählen. Den anderen schalten wir hingegen auf One-Shot (Zeile 9050), einen »Einzelschuß«. Bei einem Unterlauf lädt er zwar wieder den Startwert, bleibt aber stehen, wodurch sein Start/Stoppbit automatisch gelöscht wird. Wir prüfen dies, indem wir Bit 0 logisch nach rechts ins Carry verschieben, wo es bequem mit einem Branchbefehl untersucht werden kann. Erst der One-Shot-Betrieb des Timers B stellt sicher, daß ein Unterlauf auch erkannt wird, weil er im Dauerbetrieb vorübergehen könnte, während sich der Prozessor im Interrupt befindet. Außerdem verlangt ein gesetztes Bit 4 in der Warteschleife für beide Timer Force-Load, einen unbedingten Ladevorgang. Unabhängig davon, ob der Timer gerade läuft oder nicht, wird der Startwert geladen. Mit einem gelöschten Bit 4 kommt ein neuer Startwert erst dann zum Tragen, wenn er nach dem nächsten Unterlauf geladen wird.

Neben den Unterläufen der beiden Timer kann das Control-Register 56333 auf Bit 2 auch einen Interrupt erzeugen bei Übereinstimmung der Echtzeituhr 56328 bis 56331 mit einer vorgewählten Alarmzeit, auf Bit 3 durch ein volles oder leeres Schieberegister (56332) und auf Bit 4 durch den Impuls einer externen Signalquelle. Ein gesetztes Bit 7 zeigt hier an, daß mindestens eines der gesetzten Bits auch in der Maske gewählt ist, also ein Interrupt stattfindet. Das Interrupt-Flag wird aber bereits durch Lesen des Registers gelöscht.

Diese Kurzabhandlung vermag nur ansatzweise darzustellen, wie flexibel das Instrumentarium ist, das hier dem Anwender zur Verfügung steht. Speziell die zahlreichen Interruptmöglichkeiten verlangen geradezu nach Anwendung, wobei wir abschließend auf eine bisher nicht erwähnte noch zu sprechen kommen müssen. Denn das Demo-Programm läuft in einer Endlosschleife, die nicht ohne weiteres abgebrochen werden kann. Außerdem wird ja durch den lahmgelegten Systeminterrupt ohnehin kein Tastendruck mehr erkannt. Den Ausweg aus diesem Dilemma, haben die Konstrukteure geschaffen, als sie den sogenannten NMI erfanden. Das ist die Abkürzung für nicht maskierbarer Interrupt, eine hardwareseitige Unterbrechungsmöglichkeit mit so hoher Priorität, daß selbst ein gesetztes Interruptflag keine Rolle spielt. Wir können den NMI auslösen, indem wir die STOP-Taste gedrückt halten und gleichzeitig auf die RESTORE-Taste klopfen. Und schon ist alles wieder normal: Bild und Interrupt. Sie können zwar gleich wieder mit SYS 49152 ins Demo-Programm starten, doch vielleicht lassen Sie sich selbst einmal etwas einfallen. Nur zu — unterbrechen Sie doch Ihren Commodore 64 bitte mal ...

(Helmut Welke/aa)

Die Symboltabelle für das Maschinensprache-Programm »Rasterzeilen-Interrupt«

CHRCOL	0286	CLEAR	E544
COLOR	C198	CRA	DC0E
CRB	DC0F	EXIT	C147
FLAG	00FB	GETPAR	C17D
GOLD	C141	GRUND	D021
HIGH	C1B8	ICR	DC0D
INIT	C000	INTRO	C1C8
IREQU	D019	IRMASK	D01A
IRQ1	C12C	IRQ2	C158
IRQ3	C173	IRQOFF	EA81
LINE	00FD	LOOP1	C03E
LOOP2	C04C	LOOP3	C0B7
LOOP4	C0F9	LOW	C1A8
NAME	C1DD	OBEN	C168
PLOT	FFF0	PRINT	AB1E
RAND	D020	RASTER	D012
ROT	C139	SCREEN	D011
TEXT	C1FE	TIME	C115
TIMERA	DC04	TIMERB	DC06
VECTOR	0314	WAIT	C125

Assemblerlisting des »Rasterzeilen-Interrupt«

```
                 R A S T E R Z E I L E N - I N T E R R U P T

                  Ein Demo-Programm fuer den COMMODORE 64

    von Helmut Welke, Moehnestrasse 26, 5760 Arnsberg 1, Telefon (02932) 23627

                        ***** ASSEMBLERLISTING *****

pass 2
30:    c000                       .opt p1,oo
                        ;
50:    c000                  *=   $c000   ;startadresse 49152
                        ;
100:   c000           clear  =    $e544   ;bildschirm loeschen
110:   c000           icr    =    $dc00+13 ;interrupt-control-register
120:   c000           irmask =    $d000+26 ;interruptmaske
130:   c000           rand   =    $d000+32 ;farbe bildrand
140:   c000           grund  =    rand+1  ;farbe bildhintergrund
150:   c000           screen =    $d000+17 ;bildschirm-steuerregister
160:   c000           raster =    screen+1 ;elektronenstrahlzeile
170:   c000           vector =    $0314   ;interrupt-sprungadresse
180:   c000           timera =    $dc00+4 ;16-bit-zaehler
190:   c000           timerb =    timera+2 ;dito
200:   c000           flag   =    $fb     ;freie speicherstelle
210:   c000           cra    =    icr+1   ;steuerregister timer a
220:   c000           crb    =    cra+1   ;steuerregister timer b
230:   c000           line   =    $fd     ;freie speicherstelle
240:   c000           irequ  =    irmask-1 ;interrupt-register
250:   c000           irqoff =    $ea81   ;rueckkehr vom interrupt
260:   c000           plot   =    $fff0   ;cursor setzen/holen
270:   c000           print  =    $ab1e   ;string ausgeben
280:   c000           chrcol =    $0286   ;aktuelle zeichenfarbe
                        ;
                        ;*** programmvorbereitung ***
                        ;
1000:  c000 20 44 e5 init     jsr  clear
1010:  c003 a9 7f             lda  #%01111111 ;alle interruptmoeglichkeiten
1020:  c005 8d 0d dc          sta  icr     ;loeschen
1030:  c008 8d 04 dc          sta  timera  ;zaehlwerk auf
1040:  c00b 8d 05 dc          sta  timera+1 ;32639 einstellen
1050:  c00e a2 00             ldx  #0
1060:  c010 8e 1a d0          stx  irmask  ;kein vic-interrupt
1070:  c013 ca                dex
1080:  c014 86 fb             stx  flag    ;zaehler
1090:  c016 a9 0b             lda  #11
1100:  c018 8d 11 d0          sta  screen  ;bildschirm ausblenden
1110:  c01b 85 fd             sta  line    ;zwischenspeicher zeile
1120:  c01d a9 0e             lda  #14     ;hellblaue farbe
1130:  c01f 8d 21 d0          sta  grund
1140:  c022 85 fe             sta  line+1  ;wie 1110
                        ;
                        ;*** interrupt-beispiel bundesflagge ***
                        ;
1150:  c024 a9 2c             lda  #<irq1  ;zeiger
1160:  c026 a0 c1             ldy  #>irq1  ;auf
1170:  c028 8d 14 03          sta  vector  ;neue
1180:  c02b 8c 15 03          sty  vector+1 ;interrupt-routine
1190:  c02e a0 01             ldy  #1
1200:  c030 8c 12 d0          sty  raster  ;irr-ausloeser rasterzeile 1
1210:  c033 8c 1a d0          sty  irmask  ;irr-maske auf rasterzeilen
1220:  c036 88                dey
1230:  c037 a9 19             lda  #25
1240:  c039 20 15 c1          jsr  time    ;zeitverzoegerung
1250:  c03c a6 fe             ldx  line+1
1260:  c03e a9 02    loop1    lda  #2
1270:  c040 20 15 c1          jsr  time
1280:  c043 e8                inx
1290:  c044 86 fe             stx  line+1  ;eine zeile weiter
1300:  c046 e0 c8             cpx  #200    ;bis zeile 200
1310:  c048 90 f4             bcc  loop1   ;kleiner - dann weiter
1320:  c04a a6 fd             ldx  line    ;dito obere zeile
1330:  c04c a9 02    loop2    lda  #2
1340:  c04e 20 15 c1          jsr  time
1350:  c051 e8                inx
1360:  c052 86 fd             stx  line
1370:  c054 e0 73             cpx  #115
1380:  c056 90 f4             bcc  loop2
                        ;
                        ;*** titel-einblendung ***
                        ;
1390:  c058 a2 07             ldx  #7
1400:  c05a a0 0d             ldy  #13
1410:  c05c 18                clc
1420:  c05d 20 f0 ff          jsr  plot
1430:  c060 a9 c8             lda  #<intro ;stringadresse low
1440:  c062 a0 c1             ldy  #>intro ;und high
1450:  c064 20 1e ab          jsr  print   ;titel drucken
1460:  c067 a9 96             lda  #150
1470:  c069 a0 00             ldy  #0
1480:  c06b 20 15 c1          jsr  time
1490:  c06e a9 1b             lda  #27
1500:  c070 8d 11 d0          sta  screen  ;bildschirm oeffnen
1510:  c073 a9 c8             lda  #200
1520:  c075 20 15 c1          jsr  time
                        ;
                        ;*** interrupt-beispiel gleitzeile ***
                        ;
1530:  c078 8c 1a d0          sty  irmask  ;interrupt aus
1540:  c07b 8c 20 d0          sty  rand    ;schwarze umrandung
1550:  c07e 20 44 e5          jsr  clear
1560:  c081 a2 04             ldx  #4      ;farbe purpur
1570:  c083 8e 21 d0          stx  grund
1580:  c086 a0 03             ldy  #3
1590:  c088 18                clc
1600:  c089 20 f0 ff          jsr  plot
1610:  c08c a9 fe             lda  #<text
1620:  c08e a0 c1             ldy  #>text
1630:  c090 20 1e ab          jsr  print
1640:  c093 a9 1e             lda  #30
1650:  c095 a0 00             ldy  #0
1660:  c097 20 15 c1          jsr  time
1670:  c09a a9 58             lda  #<irq2
1680:  c09c a0 c1             ldy  #>irq2
1690:  c09e 8d 14 03          sta  vector  ;neue
1700:  c0a1 8c 15 03          sty  vector+1 ;interrupt-routine
1710:  c0a4 a2 19             ldx  #25
1720:  c0a6 86 fd             stx  line
1730:  c0a8 8e 12 d0          stx  raster
1740:  c0ab ad 19 d0          lda  irequ
1750:  c0ae 8d 19 d0          sta  irequ   ;interrupt-flag loeschen
1760:  c0b1 a0 01             ldy  #1
1770:  c0b3 8c 1a d0          sty  irmask
1780:  c0b6 88                dey
1790:  c0b7 a9 02    loop3    lda  #2
1800:  c0b9 20 15 c1          jsr  time
1810:  c0bc e8                inx
1820:  c0bd 86 fd             stx  line    ;eine zeile weiter
1830:  c0bf e0 aa             cpx  #170
1840:  c0c1 90 f4             bcc  loop3
1850:  c0c3 a9 64             lda  #100
1860:  c0c5 20 15 c1          jsr  time
                        ;
                        ;*** interrupt-beispiel regenbogen ***
                        ;
1870:  c0c8 20 44 e5          jsr  clear
1880:  c0cb 8c 1a d0          sty  irmask
1890:  c0ce a9 73             lda  #<irq3
1900:  c0d0 a0 c1             ldy  #>irq3
1910:  c0d2 8d 14 03          sta  vector
1920:  c0d5 8c 15 03          sty  vector+1
1930:  c0d8 ad 19 d0          lda  irequ
1940:  c0db 8d 19 d0          sta  irequ
1950:  c0de a0 00             ldy  #0
1960:  c0e0 84 fb             sty  flag
1970:  c0e2 c8                iny
1980:  c0e3 8c 12 d0          sty  raster
1990:  c0e6 8c 1a d0          sty  irmask
2000:  c0e9 a9 00             lda  #0
2010:  c0eb 20 15 c1          jsr  time
2020:  c0ee a9 0f             lda  #15     ;hellgraue
2030:  c0f0 8d 86 02          sta  chrcol  ;zeichenfarbe
2040:  c0f3 a2 03             ldx  #3
2050:  c0f5 18                clc
2060:  c0f6 20 f0 ff          jsr  plot
2070:  c0f9 a9 dd    loop4    lda  #<name
2080:  c0fb a0 c1             ldy  #>name
2090:  c0fd 20 1e ab          jsr  print   ;text ausgeben
2100:  c100 a9 14             lda  #20
2110:  c102 a0 00             ldy  #0
2120:  c104 20 15 c1          jsr  time
2130:  c107 ce 86 02          dec  chrcol  ;farbe - 1
2140:  c10a 10 ed             bpl  loop4   ;weiter bis einschl. 0 (=schwarz)
2150:  c10c a9 01             lda  #1
2160:  c10e a8                tay
2170:  c10f 20 15 c1          jsr  time
2180:  c112 4c 00 c0          jmp  init    ;auf ein neues
                        ;
                        ;*** warteschleife ***
                        ;
9010:  c115 8d 06 dc time     sta  timerb  ;zaehler low
9020:  c118 8c 07 dc          sty  timerb+1 ;und high laden
9030:  c11b a9 11             lda  #%00010001 ;force-load/continuous/systemtakt
9040:  c11d 8d 0e dc          sta  cra     ;timer a starten
9050:  c120 a9 59             lda  #%01011001 ;force-load/one-shot/takt timer a
9060:  c122 8d 0f dc          sta  crb     ;timer b starten
9070:  c125 ad 0f dc wait     lda  crb     ;b-control lesen
9080:  c128 4a                lsr          ;bit 0 ins carry schieben
9090:  c129 b0 fa             bcs  wait    ;timer b laeuft noch
9100:  c12b 60                rts          ;zurueck wenn aus
                        ;
                        ;*** interrupt-routinen ***
                        ;
10010: c12c a5 fb    irq1     lda  flag    ;sektorenflag
10020: c12e f0 09             beq  rot
10030: c130 10 0f             bpl  gold
10040: c132 a9 00             lda  #0
10050: c134 aa                tax
10060: c135 a4 fd             ldy  line
10070: c137 d0 0e             bne  exit
10100: c139 a9 02    rot      lda  #2
10110: c13b a2 01             ldx  #1
10120: c13d a4 fe             ldy  line+1
10130: c13f d0 06             bne  exit
10200: c141 a9 07    gold     lda  #7
10210: c143 a2 ff             ldx  #255
10220: c145 a0 01             ldy  #1
10500: c147 86 fb    exit     stx  flag    ;zeiger auf sektor
10510: c149 8c 12 d0          sty  raster  ;naechste interruptzeile
10520: c14c 8d 20 d0          sta  rand    ;farbe setzen
10530: c14f ad 19 d0          lda  irequ
10540: c152 8d 19 d0          sta  irequ   ;irq-flag loeschen
10550: c155 4c 81 ea          jmp  irqoff  ;und aussprung
                        ;
11000: [illegible] ad 21 d0 irq2     lda  grund
11010: c15b 49 02             eor  #%00000010 ;farb-flipflop 4/6
11020: c15d a8                tay
11030: c15e 29 02             and  #%00000010
11040: c160 f0 06             beq  oben    ;purpur an der reihe
11050: c162 a5 fd             lda  line
11060: c164 18                clc
11070: c165 69 15             adc  #21     ;+ 21 rasterzeilen
11080: c167 2c                .byte$2c     ;= bit (versteckter ladebefehl)
11090: c168 a5 fd    oben     lda  line
11100: c16a 8d 12 d0          sta  raster  ;zeile und
11110: c16d 8c 21 d0          sty  grund   ;farbe setzen
11120: c170 4c 4f c1          jmp  exit+8  ;aussprung
                        ;
12000: c173 c6 fb    irq3     dec  flag
12010: c175 a6 fb             ldx  flag    ;offset fuer tabellen
12020: c177 10 04             bpl  getpar
12030: c179 a2 0f             ldx  #low-color-1
12040: c17b 86 fb             stx  flag    ;offset maximal
12050: c17d bd 98 c1 getpar   lda  color,x ;farbe holen
12060: c180 48                pha          ;zunaechst auf stapel
12070: c181 bc b8 c1          ldy  high,x  ;rasterzeile highbyte
12080: c184 bd a8 c1          lda  low,x   ;und lowbyte
12090: c187 aa                tax
12100: c188 68                pla          ;farbe zurueckholen
12110: c189 8e 12 d0          stx  raster  ;zeile low
12120: c18c 8c 11 d0          sty  screen  ;und high
12130: c18f 8d 20 d0          sta  rand    ;und randfarbe und
12140: c192 8d 21 d0          sta  grund   ;hintergrundfarbe setzen
12150: c195 4c 4f c1          jmp  exit+8  ;fertig
                        ;
                        ;*** interrupt-tabellen ***
                        ;
15000: c198 04 0b 05 color    .byte4,11,5,9
15010: c19c 03 0a 01          .byte3,10,1,14
15020: c1a0 07 0f 02          .byte7,15,2,13
15030: c1a4 00 08 06          .byte0,8,6,12
                        ;
16000: c1a8 01 0b fe low      .byte1,11,254,235
16010: c1ac db cb b3          .byte219,203,179,162
16020: c1b0 8a 7a 6a          .byte138,122,106,82
16030: c1b4 4b 3b 33          .byte75,59,51,33
                        ;
17000: c1b8 1b 9b 1b high     .byte27,155,27,27
17010: c1bc 1b 1b 1b          .byte27,27,27,27
17020: c1c0 1b 1b 1b          .byte27,27,27,27
17030: c1c4 1b 1b 1b          .byte27,27,27,27
                        ;
                        ;*** tabellen hauptprogramm ***
                        ;
20000: c1c8 0e 08 1f intro    .byte14,8,31
20010: c1cb c4 45 4d          .asc "Demo-Programm"
20020: c1d8 11 11 11          .byte17,17,17,17,17
20030: c1dd 0d 1d 1d name     .byte13,29,29,29,29
20040: c1e2 1d 1d 1d          .byte29,29,29,29,29
20050: c1e7 d2 c1 d3          .asc "RASTERZEILEN-Interrupt"
20060: c1fd 00                .byte0
                        ;
21000: c1fe 9c       text     .byte156
21010: c1ff d5 4e 44          .asc "Und nun    "
21020: c20a 11 11 11          .byte17,17,17
21030: c20d 53 45 48          .asc "sehen Sie    "
21040: c21a 11 11 11          .byte17,17,17
21050: c21d 4e 4f 43          .asc "noch den"
21060: c225 0d 11 11          .byte13,17,17,17,17,17
21070: c22b 1d 1d 1d          .byte29,29,29,29,29
21080: c230 1d 1d 1d          .byte29,29,29,29,29
21090: c235 d2 a0 c5          .asc "R E G E N B O G E N"
21100: c248 00                .byte0

end of assembly $c000 - $c248

no errors
```

Betriebssystem-Erweiterung für den VC 20

Hier sind einige Erweiterungen für das Betriebssystem des VC 20, die auf Tastendruck funktionieren, wie zum Beispiel Hardcopy, Find, Relocate, Append und Beep.

Diese nützlichen Routinen werden mit dem Basic-Lader (Listing) ab Adresse $6050 in den Speicher gebracht. Ein voll ausgebauter VC 20 (+24 KByte) ist daher Voraussetzung. Um das Programm in die Tastaturabfrage einzubinden, die alle 60stel Sekunde erfolgt, wird der Zeiger für die Tastaturdecodierung (655,656) abgeändert. Damit dort nicht nach jedem STOP/RESTORE wieder der alte Wert steht, wird auch noch der BRK-Vektor (790,791) abgeändert. Zu alledem dient der SYS-Befehl in Zeile 120. Im einzelnen stehen folgende Erweiterungen zur Verfügung (die Druckroutinen sind für den Epson RX-80 mit VC-Interface):

CTRL G: Grafik auf Drucker (mit VIC 1211A)
CTRL D: Bildschirm-Hardcopy mit Zwischenzeile
CTRL H: Bildschirm-Hardcopy ohne Zwischenzeile (zum Beispiel für normale Blockgrafik)

Dabei ist zu beachten, daß das letzte Zeichen rechts unten nicht gedruckt wird, da sonst der Bildschirm gescrollt würde. Ein Bildschirm-Dump kann jederzeit erfolgen, auch während des Programmablaufs.

CTRL F: FIND mit Eingabe des Suchstrings
CTRL L: FIND weiter

FIND durchsucht ein Basic-Programm nach bestimmten Befehlen oder Zeichenfolgen und listet die Zeile, in der sich der gesuchte Begriff befindet. FIND weiter (CTRL L)bedeutet, daß das Programm weiter nach demselben Begriff durchsucht wird. Dabei ist eine Zeichenkette in Anführungszeichen einzugeben. Beispiel: Gesucht wird die Anweisung IF. Eingabe IF. Gesucht wird der String »REIF«. Eingabe »REIF«.

Im folgenden bedeutet »CTRL+Com«, daß die CTRL- und die Commodore-Taste gleichzeitig gedrückt werden.

CTRL+Com R:RELOCATE; rückgängig machen des NEW-Befehls. Diese Routine ist besonders nützlich, wenn ein Reset-Schalter existiert, da bei einem Reset ein Basic-Programm nicht zerstört wird, sondern wie bei NEW nur die ersten beiden Bytes auf Null gesetzt werden. Falls der Computer sich also mal beim Aufruf eines fehlerhaften Maschinenprogramms »aufhängt«, kann nach Reset, SYS 25600 und Drücken von CTRL+Com R ohne Verlust des Basic-Programms weitergemacht werden.

Auch nach einem »LOAD-ERROR« kann nach RELOCATE das falsch geladene Programm gelistet werden.

CTRL+Com S: Kopiert die Zeiger auf den Basic-Programmstart, die Variablen, Arrays und Strings und die ersten 65 Byte des Basic-Programms in den (hoffentlich) geschützten Bereich, in dem sich die Betriebssystemerweiterung befindet.

CTRL+Com+RETURN-Taste: Umkehrfunktion von CTRL +Com S

Falls ein Programm gelöscht und anschließend im Direktmodus mit Variablen gearbeitet wurde, dann ist der Anfang des gelöschten Programms zerstört. Das Programm kann nicht mit RELOCATE repariert werden. Wurde aber vorher zu irgendeinem Zeitpunkt, an dem das zerstörte Programm noch existierte, CTRL+Com S gedrückt, dann steht das Programm nach Betätigen von CTRL+COM+RETURN-Taste wieder mit allen Variablen, Arrays und Strings so zur Verfügung, wie zum Zeitpunkt des Kopierens.

CTRL+Com —: BEEP aus
CTRL+Com +: BEEP an

BEEP dient als akustische Rückmeldung der Tastatur, daß eine Taste gedrückt wurde. Ein momentan laufender Sound wird nicht beeinflußt.

CTRL+Com ↑: APPEND. Der Zeiger auf den Basic-Anfang wird auf das Ende des momentanen Programms gesetzt. So können mehrere Basic-Programme aneinandergehängt werden. Dabei wird die Größe des zur Verfügung stehenden Speicherplatzes ausgegeben. Durch CTRL+Com ←: werden das angehängte und das vorherige Programm verknüpft.

CTRL+Com W: WARTE. »Friert« den Computer ein, bis die RETURN-Taste gedrückt wird. Diese Routine ist vor allem bei LIST nützlich: Das Bildschirm-Scrollen wird verhindert.

Die Funktion der Zusatztasten wird dabei nicht beeinflußt, so daß während dieses Zustands zum Beispiel auch eine Hardcopy angefertigt werden kann.

Soll das Progamm mit der VC1211A-Supererweiterung zusammmenlaufen, dann müssen folgende Änderungen vorgenommen werden:

Zeile 25: Die letzten drei DATA-Werte lauten 55,163,0.
Zeile 43: Die ersten vier DATA-Werte lauten 0,32,13,164.
Zeile 120: SYS 25453 (muß auch beim Aufruf der Relocate-Funktion angegeben werden.

Nachdem diese Änderungen durchgeführt sind, stimmen natürlich die Prüfsummenabfragen nicht mehr. Es empfiehlt sich daher, das Programm in jedem Falle zunächst einmal probeweise unverändert (und ohne VC 1211A) laufen zu lassen, um eventuelle Tippfehler in den DATA-Zeilen herauszufinden. Danach kann man die Prüfsummenabfragen in den Zeilen 10, 30, 40, 60, 70, 90, 100 und 110 einfach löschen.

(Manfred Weigt/ev)

Listing. Basic-Lader zur Betriebssystem-Erweiterung

```
0 REM BETRIEBSSYSTEM-ERWEITERUNG
1 REM FUER VC 20 +24 KBYTE RAM
2 POKE55,255:POKE56,95:F$="DATA FEHLER
9 REM***TASTEN***
10 PS=0:FORT=24656TO24938:READD:POKET,D:
PS=PS+D:NEXT:IFPS<>34724THENPRINTF$:STOP
11 DATA32,211, 96,165,203,197,3,240,54,1
33,3,173,141,2,201,4,208,48,165,197
12 DATA201,19,208,3,76,110, 98,201,18,20
8,7,169,1,133,191,76,212, 98,201,43
13 DATA208,7,169,0,133,191,76,212, 98,20
1,42,208,3,76,199, 97,201,21,208,3
14 DATA76, 79, 98,76,220,235,201,6,208,2
49,165,197,201,41,208,22,162,13,181
15 DATA43,157,000,096,202,16,248,160,65,
177,43,153,14 ,96 ,136,16,248,48,221
16 DATA201,15,208,22,162,13,189,000,096,
149,43,202,16,248,160,65,185,014,096
17 DATA145,43,136,16,248,48,195,201,10,2
08,46,76,107, 97,165,203,197,3,240
18 DATA36,173,12,144,72,173,14,144,72,16
9,247,141,12,144,169,10,141,14,144
19 DATA162,255,160,10,202,208,253,136,20
8,250,104,141,14,144,104,141,12
20 DATA144,96,201,61,208,7,169, 83,141,1
43,2,208,134,201,5,208,7,169,080,141
21 DATA143,2,208,243,201,54,208,38,172,2
24,2,165,43,153,225,2,200,165,44
22 DATA153,225,2,165,45,56,233,2,133,43,
166,46,176,1,202,134,44,200,140
23 DATA224,2,32,18,228,76,116,196,201,8,
208,23,172,224,2,240,15,136,185
24 DATA225,2,133,44,136,185,225,2,133,43
,140,224,2,76,116,196,201,9,208
25 DATA10,32,159,255,32,228,255,201,13,2
08,246,76,220,235,0
29 REM***RELOCATE***
30 PS=0:FORT=24939TO25030:READD:POKET,D:
PS=PS+D:NEXT:IFPS<>12881THENPRINTF$:STOP
31 DATA165,43,133,146,133,148,165,44,133
,147,133,149,160,5,177,146,240,7
32 DATA200,208,249,230,147,208,245,200,1
52,24,101,146,133,146,144,2,230
33 DATA147,160,0,165,146,145,148,200,165
,147,145,148,165,146,133,148,165
34 DATA147,133,149,160,0,177,146,208,207
,200,177,146,208,202,200,152,208
35 DATA2,230,147,24,101,146,144,2,230,14
7,133,45,165,147,133,46,32,89,198
36 DATA76,116,196,0
39 REM***FIND***
40 PS=0:FORT=25031TO25197:READD:POKET,D:
PS=PS+D:NEXT:IFPS<>22108THENPRINTF$:STOP
41 DATA160,6,185,136,099,32,210,255,136,
208,247,162,0,32,15,225,201,13,240
42 DATA13,157,0,2,232,224,89,144,241,162
,23,76,55,196,32,202,202,162,0,160
43 DATA0,32,130,197,162,0,189,252,1,157,
61,3,240,3,232,208,245,134,187,160
44 DATA1,177,43,153,176,0,136,16,248,165
,43,133,182,165,44,133,183,160,1
45 DATA177,182,240,81,177,182,153,176,0,
136,16,248,160,4,132,188,177,182
46 DATA240,38,162,0,169,64,221,61,3,240,
14,169,34,221,61,3,240,7,177,182
47 DATA221,61,3,208,10,232,200,228,187,2
40,19,177,182,208,225,164,188,200
48 DATA208,212,165,176,133,182,165,177,1
33,183,208,186,160,3,177,182,133
49 DATA21,136,177,182,133,20,32,19,198,7
6,189,198,76,116,196,0
59 REM***GRAFIK AUF DRUCKER***
60 PS=0:FORT=25198TO25299:READD:POKET,D:
PS=PS+D:NEXT:IFPS<>14244THENPRINTF$:STOP
61 DATA169,0,32,189,255,169,4,162,4,160,
1,32,186,255,32,192,255,162,4,32
62 DATA201,255,160,3,185,143,099,32,210,
255,136,208,247,136,132,178,169,15
63 DATA133,179,162,0,160,160,169,27,32,2
10,255,169,75,32,210,255,169,160
64 DATA32,210,255,169,0,32,210,255,177,1
78,32,210,255,136,208,248,169,13
65 DATA32,210,255,165,178,24,105,160,133
,178,165,179,105,0,133,179,232,224
66 DATA20,208,203,169,4,32,195,255,96,0
69 REM***BILDSCHIRM AUF DRUCKER***
70 PS=0:FORT=25300TO25452:READD:POKET,D:
PS=PS+D:NEXT:IFPS<>20321THENPRINTF$:STOP
71 DATA8,72,138,72,152,72,186,134,166,56
,32,240,255,134,167,132,168,169
72 DATA0,32,189,255,169,4,162,4,160,0,32
,186,255,32,192,255,32,129,229,169
73 DATA0,32,189,255,169,3,162,3,160,0,32
,186,255,32,192,255,162,4,32,201
74 DATA255,162,3,32,198,255,162,22,160,2
2,169,255,133,205,32,207,255,201
75 DATA13,240,3,32,210,255,136,208,239,1
65,191,208,5,169,8,32,210,255,169
76 DATA13,32,210,255,169,15,32,210,255,2
02,240,41,16,213,166,167,164,168
77 DATA24,32,240,255,169,13,32,210,255,1
69,3,133,154,32,195,255,169,4,32
78 DATA195,255,169,0,133,153,166,166,154
,104,168,104,170,104,40,96,160,21
79 DATA208,172,0
89 REM***EINBINDEN INS BETRIEBSSYSTEM WE
NN VIC1211A ANGESCHLOSSEN IST***
90 PS=0:FORT=25453TO25479:READD:POKET,D:
PS=PS+D:NEXT:IFPS<>2498THENPRINTF$:STOP
91 DATA32,056,162,169,080,160,096,141,14
3,2,140,144,2,169,109,160,099,141,22
92 DATA3,140,23,3,108,2,192,0
99 REM***EINBINDEN INS BETRIEBSSYSTEM OH
NE ROM-ZUSATZ***
100 PS=0:FORT=25600TO25631:READD:POKET,D
:PS=PS+D:NEXT:IFPS<>3326THENPRINTF$:STOP
101 DATA32,82,253,32,249,253,32,24,229,1
69,080,160,096,141,143,2,140,144,2
102 DATA169,000,160,100,141,22,3,140,23,
3,108,2,192
109 REM***' ?DNIF' UND DRUCKANWEISUNGEN*
**
110 PS=0:FORT=25480TO25490:READD:POKET,D
:PS=PS+D:NEXT:IFPS<>994THENPRINTF$:STOP
111 DATA255,32,63,68,78,73,70,255,8,65,2
7
120 SYS25600
READY.
```

Befehlserweiterung für Simons Basic

Die Fähigkeiten des Commodore 64 sind mit dem vorhandenen Befehlsvorrat des Basic 2.0 nur sehr schwer auszunutzen. Dafür bietet Commodore eine Erweiterung an, die diesen Mangel weitgehend behebt: Simons Basic. Dieser Artikel gibt eine grobe Speicherbelegung und behebt einige Mängel.

Leider muß im voraus erwähnt werden, daß nur Besitzer der Disk-Version POKEs anwenden können, da bei der Modul-Version ein unveränderbares ROM vorliegt und somit POKEs hier unwirksam sind.

Obwohl nur 8 KByte im Basic-Speicher verbraucht werden, ist Simons Basic eine 16-KByte-Basic-Erweiterung. Die zweiten 8 KByte liegen unter dem Basic-ROM im RAM, so daß Simons Basic immer trickreich zwischen beiden Ebenen umschalten muß. Zusätzlich wird auch der Bereich von $C400-$CBFF (dezimal 50176-52223) benutzt; zum Beispiel liegt der KEY-Funktionstastenspeicher ab $C64D (50765).

Bei MEM (dem Befehl zum Kopieren des Original-Zeichen-ROMs ins RAM und dessen Einschalten) liegt der Zeichensatz ab $E000 (57344) im RAM unter dem Kernal, der Bildschirm von nun an ab $CC00 (52224), die Sprite-Pointer ab $CFF8 (53240) und die Sprites von $C000-$C3FF (49152-50175).

Bei HIRES und MULTI (den Grafik- und Farbgrafik-Modi) liegt der Grafik-Speicher unter dem Kernal im RAM ($E000-$FFFF, 57344-65535), der Farbspeicher ab $C000 (49152), und Sprites finden sich in den Blocks 48-63 (ab $CC00, 52224).

Man sieht also, daß $C000-CFFF (49152-53247) laufend belegt sind und kleine Maschinenroutinen höchstens in Sprite-Speichern plaziert werden können.

Hier erst einmal Beanstandungen zur Beschreibung einiger Befehle:

```
50 REM **********************************
51 REM *            D  U  M  P          *
52 REM **********************************
53 REM * BERICHTIGUNG DES               *
54 REM * SIMON'S BASIC-BEFEHLS DUMP     *
55 REM **********************************
59 K=0:PRINT"▒▒ D U M P ▒     ?CHECKSUM "
;
60 FOR I=$9F9C TO $9FA7:READ J:POKE I,J:
K=K+J:NEXT
61 DATA 160,4,32,57,160,136,208,250,234
62 DATA 234,234,234:POKE $80B9,221:POKE
$808C,68
63 POKE $9FD6,240:POKE $9FD7,27:POKE $9F
D8,200
64 FOR I=$9FF0 TO $9FFC:READ J:POKE I,J:
K=K+J:NEXT
65 DATA 160,4,44,160,5,169,34,32,210,255
,76,158,159
66 IFK=3409THENPRINT"OK":ELSE:PRINT"ERRO
R
READY.
```

Listing 1. Variablen-Liste mit Vorzeichen

```
70 REM **********************************
71 REM *            P  A  G  E          *
72 REM **********************************
73 REM * BERICHTIGUNG DES               *
74 REM * SIMON'S BASIC-BEFEHLS PAGE     *
75 REM **********************************
79 K=0:PRINT"▒▒ P A G E ▒     ?CHECKSUM "
;
80 FOR I=$BFF2 TO $BFFE:READ J:POKE I,J:
K=K+J:NEXT
81 DATA 32,225,255,208,5,104,104,76,60
82 DATA 191,76,228,255
83 POKE $BF12,242:POKE $BF13,191
84 IFK=1819THENPRINT"OK":ELSE:PRINT"ERRO
R
READY.
```

Listing 2. Verbesserte Version des Page-Befehls

```
10 REM ******************************
11 REM *                            *
12 REM *  INTERPRETERCODETABELLE    *
13 REM *                            *
14 REM ******************************
20 A=0:B=33762
30 A=A+1:IF A=128 THEN B=41118
40 IF A=204 THEN END
50 PRINT:PRINT A,
60 PRINT CHR$(PEEK(B) AND 127);:B=B+1
61 IF A<128 AND PEEK(B)<>64 THEN 60
62 IF A>127 AND PEEK(B-1)<128 THEN 60
70 IF A<128 THEN B=B+1
80 GOTO 30
READY.
```

Listing 3. Alle Befehle des C 64 mit Simons Basic

```
10 REM ********************************
11 REM *          E  R  R  O  R       *
12 REM ********************************
13 REM * ERGAENZUNG ZU SIMON'S BASIC  *
14 REM * FORMAT: ERROR                *
15 REM * GIBT DISKSTATUS AUS          *
16 REM * > BEISPIEL: ERROR            *
17 REM *   ERGEBNIS: 00, OK,00,00     *
18 REM ********************************
19 K=0:PRINT" E R R O R   ?CHECKSUM "
;
20 FORI=$BFBB TO $BFE2:READ J:POKE I,J:K
=K+J:NEXT
21 DATA32,115,,201,176,240,3,76,138,136
22 DATA169,8,133,186,32,180,255,169,111
23 DATA133,185,32,150,255,32,165,255,32
24 DATA210,255,201,13,208,246,32,171
25 DATA255,76,116,145
26 POKE$87BE,186:POKE$87BF,191
27 IFK=5527THENPRINT"OK":ELSE:PRINT"ERRO
R"
READY.
```

Listing 4. Disk-Status-Ausgabe

```
30 REM ********************************
31 REM *         J  O  Y              *
32 REM ********************************
33 REM * ERGAENZUNG ZU SIMON'S BASIC  *
34 REM * FORMAT: JOY N                *
35 REM * STELLT JOYSTICKPORT EIN      *
36 REM * > BEISPIEL: JOY1 (FUER PORT1)*
37 REM *             JOY2 (FUER PORT2)*
38 REM ********************************
39 K=0:PRINT" J O Y        ?CHECKSUM "
;
40 FORI=$BFE3 TO $BFF1:READ J:POKE I,J:K
=K+J:NEXT
41 DATA32,115,,32,6,129,138,41,1,141
42 DATA213,136,76,42,130
43 POKE$88DA,111:POKE$88FC,16
44 POKE$8764,226:POKE$8765,191
45 IFK=1232THENPRINT"OK":ELSE:PRINT"ERRO
R"
READY.
```

Listing 5. Joystick für beide Ports

Beim **TEXT**-Befehl (Einsetzen von Text in die HiRes-Grafik-Seite) müssen Zeichentyp, Größe und Abstand als Konstanten gegeben sein, da der Text sonst nicht korrekt ausgedruckt wird. <CTRL-A> und <CTRL-B> können beliebig in einem String benutzt werden. Kein CTRL am Anfang wird als Großschrift ausgelegt. <RVS ON/OFF>-Zeichen werden auch richtig ausgeführt.

Beim Befehl **FETCH** (Eingaberoutine mit bestimmbarer Zeicheneingabe-Beschränkung) fehlt die Angabe, daß die Eingabelänge über ein Zeichen (wie angegeben) hinausgehen darf. Das Limit liegt bei 88 Zeichen. Danach ist der Eingabepuffer, in den die eingegebenen Zeichen abgelegt werden, gefüllt und dahinterliegende Systemvariablen (siehe C 64-Handbuch) können zerstört werden. Ein Fehler ist außerdem, daß nach der Eingabe des letzten Zeichens keine Korrektur (mit <DEL>) mehr möglich ist. Es bleibt keine andere Wahl, als <RETURN> zu drücken.

Bei **DUMP** (Ausgabe der Inhalte aller nicht indizierten Variablen) werden die Werte leider ohne Vorzeichen ausgegeben. Dies kann man beheben mit **POKE 32953,221**. Zudem werden leere Strings (Länge 0) als Strings mit zufälligen 255 Zeichen und Integervariablen als 16-Bit-Adressen ausgegeben. Das Programm in Listing 1 beseitigt alle drei Fehler.

Im Handbuch wird bei den Bildschirmroll-Befehlen das Format leider falsch angegeben. Die Parameter werden mit der Routine geholt, die auch INV und MOVE bedient. Daher ist auch das Format das gleiche:
Richtung W/B r,c,w,d.

Die Parameter entsprechen denen im Simons Basic-Handbuch in Abschnitt 7.6.

Bei **MERGE** kann es vorkommen, daß ein Programm nicht ordnungsgemäß angehängt wird. Dieser Fehler kommt nicht mehr vor, wenn man immer **OLD** vor MERGE eingibt, da dieser Befehl das Programmende noch einmal überprüft und gegebenenfalls bereinigt.

Bei **PAGE** kann man das Listen nicht, wie in der Anleitung angegeben, durch Drücken der RUN/STOP-Taste abbrechen. Das Programm in Listing 2 behebt diesen Fehler.

Befehle, die zwar nicht im Handbuch, jedoch in der Befehlsliste im RAM zu finden sind, wurden bereits in früheren Ausgaben behandelt.

Zur Speicherung der Simons Basic-Befehle:

Die Befehle werden als Zwei-Byte-Kombination abgespeichert. Das erste Byte hat den Wert 100, und das zweite einen Wert zwischen 1 und 127. Dies ergäbe eine Befehlsmenge von 127 Befehlen, einige Tokens sind aber nicht belegt.

Um alle Befehlscodes auszugeben, kann man Listing 3 verwenden. Nach RUN wird eine Liste der Interpretercodes ausgegeben, die direkt aus dem RAM von Simons Basic und aus dem Basic-ROM entnommen wird. Setzt man in Zeile 20 für A den Wert 127 ein, so wird nur der normale Befehlsvorrat ausgedruckt. Mit diesem geänderten Programm kann sich jeder C 64-Anwender eine Interpretercodetabelle erstellen, die leider im C 64-Handbuch fehlt.

Nun zwei kurze Maschinenprogramme, die nach einmaligem Lauf als neue Befehle zur Verfügung stehen:

ERROR:

Ausgabe des Diskettenstatus auf dem Bildschirm (Listing 4).

JOY n:

Nach Ausführung dieses Befehls liest die Funktion JOY den Control Port n (n=1 oder 2) (Listing 5).

Die Listings 1 und 5 lassen sich zum Beispiel als Vorprogramm nach dem Start von Simons Basic laden und starten und stehen danach bis zum Ausschalten des Computers bereit.

(Dieter Temme/gk)

Die Ebenen des Absturzes

Nachdem ich mir den Commodore 64 gekauft hatte, schnappte ich mir das Handbuch und fing an, in Basic zu programmieren. Wenn ich einmal versehentlich in eine Endlosschleife geriet, drückte ich die Run/Stop-Taste, und das Programm wurde unterbrochen, damit ich den Fehler beseitigen konnte — die Computerwelt war in Ordung!

Doch wer den C 64 kennt, hat sich sicher schon an die Befehle PEEK und POKE versucht. Zum Beispiel:

```
100 FOR I = 0 TO 999
110 POKE 1024 + I,0
120 POKE 55296+I,0
130 NEXT I
```

Dieses kleine Programm füllt den Bildschirm mit schwarzen Klammeraffen. Doch wehe, man hat in Zeile 100 statt 999 die leicht erweiterte Version 9999 stehen! Direkt hinter dem Bildschirmspeicher liegt der Basic-Benutzerspeicher. Bei solchen Fehlern frißt sich das Programm von selbst auf. Das Programm ist teilweise, wenn nicht ganz, zerstört. Der fortgeschrittene Programmierer macht sich sicher eines Tages Gedanken, wie er seine Programme durch versehentliches Drücken der Run/Stop-Taste schützen kann.

Dieses ist besonders dann angebracht, wenn das fertige Programm von einem C 64-Unkundigen bedient werden soll.

Dieses kleine Porgramm bewirkt Wunder:

```
100 FOR I=830 TO 834
110 READ A : POKE I,A : NEXT I
120 POKE 808,62 : POKE 809,3
130 DATA 169,1,201,0,96
```

Weshalb? Im Betriebssystem gibt es ein Unterprogramm, welches die Stopp-Taste abfragt. Das Programm liegt zwischen 63213 ($ F6ED) und 63226 ($ F6FA). Die Anfangsadresse dieser Routine ist in den Speicherzellen 808 und 809 gespeichert; und zwar Low-Byte vor High-Byte. Nun wird diese Routine regelmäßig aufgerufen. Ist die Stopp-Taste gedrückt worden, so veranlaßt diese Routine — abgesehen von ein paar anderen Instruktionen —, daß das Zeroflag gesetzt wird. Wurde die Stopp-Taste aber nicht gedrückt, so löscht dieses Unterprogramm das Zeroflag. Wenn die Speicherzellen 808 und 809 so verändert werden, daß zu einer Routine gesprungen wird, die immer das Zeroflag löscht, dann wird nie das Drücken der Stopp-Taste erkannt. Dieses geschieht durch unser kleines Programm, welches eine Routine in den Kassettenpuffer schreibt.

Haben wir für kritische Programmteile die Stopp-Taste unterdrückt, können wir jene wieder durch den Befehl

```
200 POKE 808,237 : POKE 809,246
```

aktivieren.

Nehmen wir einmal an, uns ist folgendes passiert: Wir haben ein Programm geschrieben, das die Stopp-Taste ausschaltet (damit auch Run/Stop + Restore). Das Programm stürzt ab (zum Beispiel durch eine Endlosschleife), und wir haben es noch nicht abgespeichert. Das Programm scheint verloren, denn wir können den Computer nur noch aus- und einschalten. Doch da gibt es noch eine Rettung. Das magische Wort heißt RESET.

Falls Sie in einem Programm einen Reset wünschen, dann benutzen Sie folgenden Befehl:

```
100 SYS 64738
```

Der Bildschirm verengt sich links und rechts um ein halbes Zeichen, und dann meldet sich der Computer, als ob sie ihn gerade eingeschaltet hätten. Hardwaremäßig bewirken sie einen Reset, indem Sie Pin 1 und Pin 3 am User-Port verbinden.

Sie können aber auch Pin 2 und Pin 6 am seriellen Bus verbinden. Dazu nehmen Sie das Kabel aus dem Floppy-Laufwerk und verbinden die beiden Pins. Das Prinzip ist, die Reset-Leitung mit der GND-Leitung (Erde) zu verbinden.

Komfortabler geht es mit einem Resetschalter, der in den User-Port eingeschoben wird und per Knopfdruck einen Reset bewirkt.

Wenn Sie nun versuchen, Ihr Programm zu listen, wird Ihre Skepsis zunächst bestätigt werden. Es gibt scheinbar kein Programm mehr.

Durch das Rücksetzen (Reset) des Computers sind alle Basic-Pointer auf ihren Urzustand gebracht worden. Das Programm selbst existiert noch, denn der Basic-Benutzerspeicher wurde nicht gelöscht. Um die Zeiger (Pointer) zurückzusetzen, und somit Ihr Programm wieder sichtbar zu machen, laden Sie das Programm »UNNEW« ein, das Sie hoffentlich vorher eingetippt haben. Nun starten Sie es und — das Basic-Programm wird wieder sichtbar.

Haben Sie vor dem Reset in Maschinensprache mit einem Monitor (zum Beispiel HES-Mon oder Supermon 64) programmiert, dann starten Sie den Monitor und fahren Sie mit der Programmierung beziehungsweise dem Testen fort. Sowohl Ihr Maschinenprogramm als auch der Monitor wurden nicht zerstört.

Haben Sie sich einmal mit dem Umgang des Resets vertraut gemacht, werden Sie sicher auch gerne einmal hinter die Kulissen von Videospielen schauen wollen. Dieses konnten Sie

bisher nicht, weil viele Programme die Run/Stop-Taste sperren. Funktioniert der Reset, dann können Sie mit einem Monitor das Spiel erforschen. Doch was geschieht, wenn der Reset wirkungslos bleibt? Wie kann sich ein Videospiel gegen das scheinbare Aus- und Einschalten schützen?

Die Lösung besteht in der Möglichkeit, Module anzuschließen, nämlich wenn Sie ein Modul einschieben und das Gerät einschalten, dann meldet sich keineswegs Basic V2.0 mit einem READY, sondern das Modul übernimmt das Kommando, ohne daß Sie es dazu aufgefordert hätten.

Dazu muß das Modul drei Sachen veranlassen:
1. Den Basic-Interpreter ausblenden,
2. sich als Modul zu erkennen geben und
3. eine Einsprungadresse zur Verfügung stellen.

Das Wegbleiben eines ROMs, sei es der Basic-Interpreter oder das Betriebssystem, erfolgt durch Setzen beziehungsweise Löschen von Bits in der Speicherzelle 0001. Dort liegen die für uns interessanten Kanäle High-RAM und Low-RAM, die durch die beiden Bits 0 und 1 dargestellt werden. Im Einschaltzustand sind beide gesetzt. Ferner gibt es die hardwaremäßigen Kanäle GAME und ExROM. Diese liegen im Moduleinschub und sind ohne Modul auf High (1) gesetzt. Wird ein Modul eingeschoben, so bewirkt dieses, daß die Kanäle GAME und ExROM auf Low gesetzt werden. Dadurch wird das Basic-ROM ausgeblendet.

Als zweites muß sich das Modul zu erkennen geben. Im Betriebssystem gibt es eine Routine, die erkennt, ob ein Modul eingeschoben ist. Sie liegt zwischen 64770 ($ FD02) und 64788 ($ FD14). Dabei wird überprüft, ob in den Speicherzellen 32772 bis 32777 ($ 8004 bis $ 8009) das Wort »cbm80« steht. Das sind die ASCII-Zeichen: 195,194,205, 56,48. Ist das der Fall, setzt das Unterprogramm das Zeroflag. Wenn nun eine Routine im Betriebssystem (zum Beispiel die Reset-Routine) dieses Unterprogramm aufruft, und das Zeroflag wird gesetzt, dann nimmt das Betriebssystem an, daß ein Modul vorliegt. Es springt zur Adresse, die in $8002 und $ 8003 steht. Die Steuerung wird dem Modul übergeben. Will man nun ein Modul simulieren, muß man
1. $ 8004 bis 8009 mit »cbm80« belegen,
2. eine Adresse in $ 8002 und $ 8003 eintragen (Low-Byte vor High-Byte) und
3. durch den Befehl CLI Interrupts (IRQ) erlauben.

Mit folgendem kleinen Programm sorgt man von Basic aus dafür, daß ein Reset keinen Effekt hat:

```
100 FOR I=32770 TO 32778
110 READ A : POKE I,A
120 NEXT I
130 DATA 10, 128, 195, 194, 205
140 DATA 56, 48, 88, 0
```

Unser Ausgangsproblem war aber: Wie kann ich ein Reset verursachen, obwohl es gesperrt ist? Es gibt zwei Methoden:
1. Man sorgt dafür, daß in $ 8004 bis $ 8009 nicht »cbm80« steht.
2. Die Routine im Betriebssystem ändert man so ab, daß sie nicht mehr »cbm80« sondern zum Beispiel »cbm81« abfragt.

Es soll zuerst der zweite Fall behandelt werden. Da das Betriebssystem (auch Kernal genannt) im ROM liegt, läßt es sich nicht ohne weiteres ändern. Deshalb wird es zuerst in das darunterliegende RAM kopiert. Dieses funktioniert folgendermaßen:

```
100 FOR I=57344 TO 65535
110 POKE I,PEEK(I)
120 NEXT I
```

Nun könnte, durch Löschen des Bits 1 (High-RAM) in der Speicherzelle 0001, vom ROM auf RAM umgeschaltet werden:

```
320 POKE I,PEEK(I) AND 253
```

Durch das Löschen dieses Bits wird aber auch das Basic-ROM ausgeblendet. Also müssen wir dieses auch ins RAM kopieren:

```
200 FOR I=40960 TO 49151
210 POKE I,PEEK(I)
220 NEXT I
```

Jetzt könnte man das Programm starten, es hätte jedoch keinen Effekt, weil in der Adresse 0001 immer die Zahl 55 (High-RAM gesetzt) erscheint — statt der angestrebten 53. Dieses kommt durch den I/O-Reset. Diese Routine liegt im Kernal und setzt die Kanäle in 0001 immer wieder neu. Also muß man diese Routine für unsere Zwecke ändern:

```
300 POKE 64982,229
```

Schließlich werden wir aus der »cbm80«-Abfrage eine »cbm81«-Abfrage machen:

```
310 POKE 64788,49
```

Wenn man jetzt das Programm startet und ein Reset auslöst durch: SYS 64738 wird sich die Maschine zurücksetzen, auch wenn in 32772 bis 32777 (exklusive) »cbm80« stehen sollte. Betätigt man jedoch den Resetschalter, dann wird der Reset abhängig von 32772 bis 32777 ausgelöst. Beim Reset werden nicht nur der Prozessor, sondern auch die beiden CIAs und der SID zurückgesetzt. Diese bewirken, daß in 0001 High-RAM gesetzt wird, und somit kommt die Veränderung des Betriebssystems nicht zum Tragen. Im ersten Fall wird ein POKE 32772,0 das Nötige tun, sofern nicht das Programm seinerseits dafür sorgt, daß »cbm80« immer wieder erneuert wird.

Hier das Programm UNNEW:

```
100 FOR I=525 TO 578
120 READ A : POKE I,A : NEXT I
200 POKE 43,525 AND 255 : POKE 44,2
210 POKE 45,578 AND 255 : POKE 46,2
220 CLR : SAVE "UNNEW" ,8 : REM bzw. ,1,1
300 DATA 160,003,200,177,043,208,251,200
310 DATA 200,152,160,000,145,043,165,044
320 DATA 200,145,043,133,060,160,000,132
330 DATA 059,162,000,200,208,002,230,060
340 DATA 177,059,208,245,232,224,003,208
350 DATA 242,200,208,002,230,060,132,045
360 DATA 164,060,132,046,096,256
```

Wenn Sie dieses Programm eingeben und starten, wird es ein Programm namens »UNNEW« auf Diskette schreiben. Falls sie aus Versehen NEW eingetippt haben, dann laden sie das Programm durch

LOAD "UNNEW" ,8,1

und starten es durch

SYS 525

Ihr Basic-Programm ist gerettet, selbst wenn sie ein Reset ausgelöst haben. Das Programm setzt die Zeiger in $ 0801 und $ 0802, die auf die nächste Zeile zeigen, auf den richtigen Wert. Bei NEW werden diese beiden Bytes auf Null gesetzt, und zwei aufeinanderfolgende Nullen bedeuten für den Interpreter »Ende des Programms«.

(Daniel Kossmann/aa)

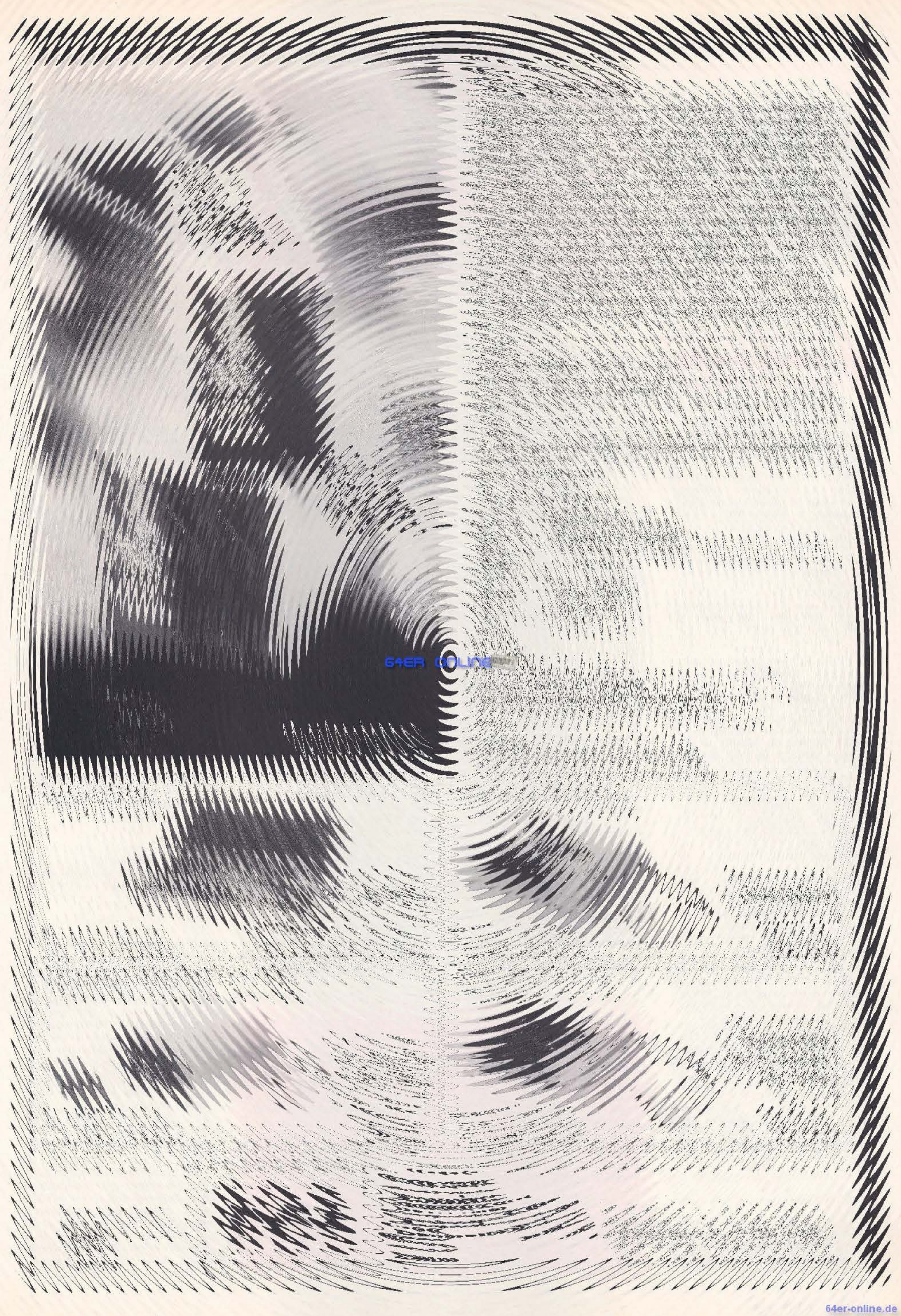
Epedemic 2

Ein Nachtrag zu dem VC 20-Spielprogramm aus der 64er/Ausgabe 10

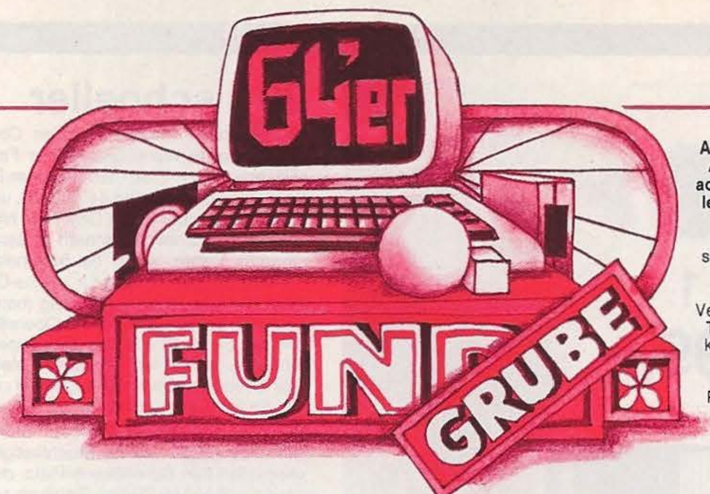
Nachdem der Druckfehlerteufel in der letzten Ausgabe wieder einmal zugeschlagen hatte, bringen wir hier den fehlenden Teil zum Listing »Epedemic« für den VC 20. Nachzutragen wäre auch noch, daß man sich bei derartigen Simulationsspielen immer über die Handlungsbrisanz im Klaren sein sollte. Denn die Zahlenjonglierereien auf dem Bildschirm stellen immerhin Menschenleben dar. Wem das Spiel allerdings deswegen zu makaber ist, der sollte sich vielleicht einmal die Realität ansehen... (ev)

```
50530 DATA4182,160,4183,
 126,4185,126,4186,251,4
 187,160,4188,160,4189,1
 60,4190,160
50540 DATA4191,123,4195,
 120,4196,225,4197,160,4
 198,160,4199,160,4200,1
 60,4201,160
50550 DATA4202,160,4203,
 160,4204,234,4208,106,4
 209,160,4210,160,4211,1
 60,4212,236
50560 DATA4216,108,4217,
 254,4218,160,4219,160,4
 220,160,4221,160,4222,1
 60,4223,160
50570 DATA4224,160,4225,
 160,4226,236,4230,245,4
 231,160,4232,160,4233,2
 36,4238,254
50580 DATA4239,160,4240,
 160,4241,160,4242,160,4
 243,160,4244,160,4245,1
 60,4246,160
50590 DATA4247,160,4252,
 160,4253,160,4254,251,4
 259,108,4260,236,4261,1
 24,4262,123
50600 DATA4263,245,4264,
 160,4265,160,4266,160,4
 267,160,4268,160,4269,1
 60,4270,116
50610 DATA4274,117,4275,
 160,4276,124,4281,124,4
 282,126,4284,126,4285,2
 54,4286,160
50620 DATA4287,160,4288,
 160,4289,160,4290,160,4
 291,236,4297,236,4303,2
 54,4304,160
50630 DATA4305,247,4306,
 123,4307,160,4308,160,4
 309,160,4310,160,4311,1
 60,4312,160
50640 DATA4313,97,4319,1
 27,4320,123,4325,160,43
 26,160,4327,160,4328,16
 0,4329,95
50650 DATA4330,105,4331,
 225,4332,97,4333,95,433
 4,236,4342,127,4343,252
 ,4344,123
50660 DATA4347,160,4348,
 160,4349,160,4350,160,4
 351,123,4353,106,4354,1
 01,4356,251
50670 DATA4364,160,4365,
 160,4366,252,4370,251,4
 371,160,4372,160,4373,9
 7
50680 DATA4378,123,4386,
 245,4387,160,4388,160,4
 389,116,4392,106,4393,1
 60,4394,160
50690 DATA4395,101,4400,
 127,4408,225,4409,160,4
 410,160,4411,101,4414,1
 03,4415,160
50700 DATA4416,160,4430,
 106,4431,160,4432,236,4
 437,251,4438,231,4445,2
 54
50710 DATA4446,254,4452,
 245,4453,160,4454,97,44
 59,225,4460,126,4467,25
 1
50720 DATA4468,236,4474,
 118,4475,236,4496,103,4
 497,117,4519,126
50730 DATA4324,106,4298,
 59,4292,118
51000 REM**************
51010 REM BER.GRAFIK
51020 REM**************
51030 IFC(1)=1THEN52000
51040 IFC(2)=1THEN52100
51050 IFC(3)=1THEN52200
51060 IFC(4)=1THEN52300
51070 IFC(5)=1THEN52400
51080 IFC(6)=1THEN52500
51090 IFC(7)=1THEN52600
51100 RETURN
52000 IFEP(1)<25THENC2=2
 2:C3=0:C4=0:C5=0:C6=0:C
 7=0:GOTO53000
52010 IFEP(1)<=50THENC2=
 22:C3=44:C4=0:C5=0:C6=0
 :C7=0:GOTO53000
52020 IFEP(1)<=75THENC2=
 22:C3=44:C4=66:C5=88:C6
 =0:C7=0:GOTO53000
52030 IFEP(1)<=100THENC2
 =22:C3=44:C4=66:C5=88:C
 6=110:C7=132:GOTO53000
52100 IFEP(2)<=25THENC2=
 22:C3=0:C4=0:C5=0:GOTO5
 4000
52110 IFEP(2)<=50THENC2=
 22:C3=44:C4=0:C5=0:GOTO
 54000
52120 IFEP(2)<=75THENC2=
 22:C3=44:C4=66:C5=0:GOT
 054000
52130 IFEP(2)<=100THENC2
 =22:C3=44:C4=66:C5=88:G
 OTO54000
52200 IFEP(3)<=25THENC2=
 22:C3=0:C4=0:C5=0:C6=0:
 GOTO55000
52210 IFEP(3)<=50THENC2=
 22:C3=44:C4=0:C5=0:C6=0
 :GOTO55000
52220 IFEP(3)<=75THENC2=
 22:C3=44:C4=66:C5=0:C6=
 0:GOTO55000
52230 IFEP(3)<=100THENC2
 =22:C3=44:C4=66:C5=88:C
 6=110:GOTO55000
52300 IFEP(4)<=25THENC2=
 22:C3=0:C4=0:C5=0:C6=0:
 C7=0:C8=0:GOTO56000
52310 IFEP(4)<=50THENC2=
 22:C3=44:C4=66:C5=0:C6=
 0:C7=0:C8=0:GOTO56000
52320 IFEP(4)<=75THENC2=
 22:C3=44:C4=66:C5=88:C6
 =110:C7=0:C8=0:GOTO5600
 0
52330 IFEP(4)<=100THENC2
 =22:C3=44:C4=66:C5=88:C
 6=110:C7=132:C8=154:GOT
 056000
52400 IFEP(5)=<25THENC2=
 0:C3=0:C4=0:GOTO57000
52410 IFEP(5)<=50THENC2=
 1:C3=0:C4=0:GOTO57000
52420 IFEP(5)<=75THENC2=
 1:C3=22:C4=0:GOTO57000
52430 IFEP(5)<=100THENC2
 =1:C3=22:C4=23:GOTO5700
 0
52500 IFEP(6)<=25THENC2=
 22:C3=0:C4=0:C5=0:C6=0:
 C7=0:C8=0:GOTO58000
52510 IFEP(6)<=50THENC2=
 22:C3=44:C4=0:C5=0:C6=0
 :C7=0:C8=0:GOTO58000
52520 IFEP(6)<=75THENC2=
 22:C3=44:C4=66:C5=88:C6
 =0:C7=0:C8=0:GOTO58000
52530 IFEP(6)<=100THENC2
 =22:C3=44:C4=66:C5=88:C
 6=110:C7=132:C8=154:GOT
 058000
52600 IFEP(7)<=25THENC2=
 22:C3=44:C4=0:C5=0:C6=0
 :C7=0:C8=0:C9=0:CC=0:GO
 TO59000
52610 IFEP(7)<=50THENC2=
 22:C3=44:C4=66:C5=88:C6
 =0:C7=0:C8=0:C9=0:CC=0:
 GOTO59000
52620 IFEP(7)<=75THENC2=
 22:C3=44:C4=66:C5=88:C6
 =110:C7=132:C8=0:C9=0:C
 C=0:GOTO59000
52630 IFEP(7)<=100THENC2
 =22:C3=44:C4=66:C5=88:C
 6=110:C7=132:C8=154:C9=
 176:CC=198
52640 GOTO59000
53000 FORC1=38056TO38062
53010 POKEC1,4:POKEC1+C2
 ,4:POKEC1+C3,4:POKEC1+C
 4,4:POKEC1+C5,4:POKEC1+
 C6,4:POKEC1+C7,4
53030 NEXT:GOTO51040
54000 FORC1=37946TO37953
54010 POKEC1,4:POKEC1+C2
 ,4:POKEC1+C3,4:POKEC1+C
 4,4:POKEC1+C5,4
54020 NEXT:GOTO51050
55000 FORC1=37963TO37967
55010 POKEC1,4:POKEC1+C2
 ,4:POKEC1+C3,4:POKEC1+C
 4,4:POKEC1+C5,4:POKEC1+
 C6,4
55020 NEXT:GOTO51060
56000 FORC1=38094TO38099
56010 POKEC1,4:POKEC1+C2
 ,4:POKEC1+C3,4:POKEC1+C
 4,4:POKEC1+C5,4:POKEC1+
 C6,4:POKEC1+C7,4
56015 POKEC1+C8,4
56020 NEXT:GOTO51070
57000 FORC1=38237TO38237
57010 POKEC1,4:POKEC1+C2
 ,4:POKEC1+C3,4:POKEC1+C
 4,4
57020 NEXT:GOTO51080
58000 FORC1=37932TO37939
58010 POKEC1,4:POKEC1+C2
 ,4:POKEC1+C3,4:POKEC1+C
 4,4:POKEC1+C5,4:POKEC1+
 C6,4:POKEC1+C7,4
58015 POKEC1+C8,4
58020 NEXT:GOTO51090
59000 FORC1=38110TO38115
```

Die fehlenden Zeilen von »Epedemic« aus der 64'er, Ausgabe 10/84


64ER ONLINE

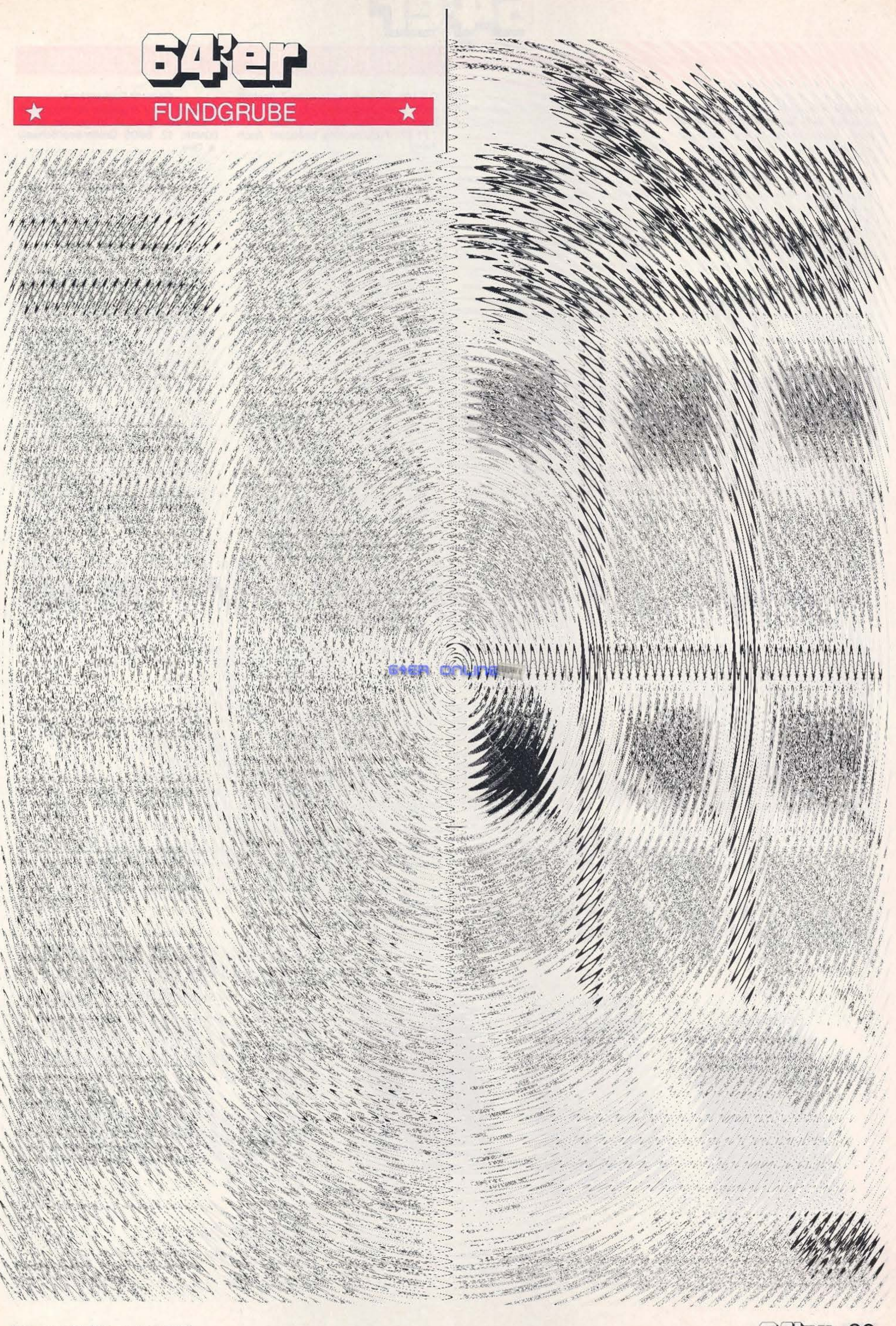
Wollen Sie einen gebrauchten Computer verkaufen oder erwerben? Suchen Sie Zubehör? Haben Sie Software anzubieten oder suchen Sie Programme oder Verbindungen? Die FUNDGRUBE von 64'er bietet allen Computerfans die Gelegenheit, für nur DM 5,— eine private Kleinanzeige mit bis zu 5 Zeilen Text in der Rubrik Ihrer Wahl aufzugeben. Und so kommt Ihre private Kleinanzeige in die FUNDGRUBE der **Dezember-Ausgabe** (erscheint am 16. November 84): Schicken Sie Ihren Anzeigentext bis zum 23. Oktober 84 (Datum des Poststempels und Anzeigenschluß) an »64'er«. Später eingehende Aufträge werden in der **Januar-Ausgabe** (erscheint am 14. Dezember 84) veröffentlicht.

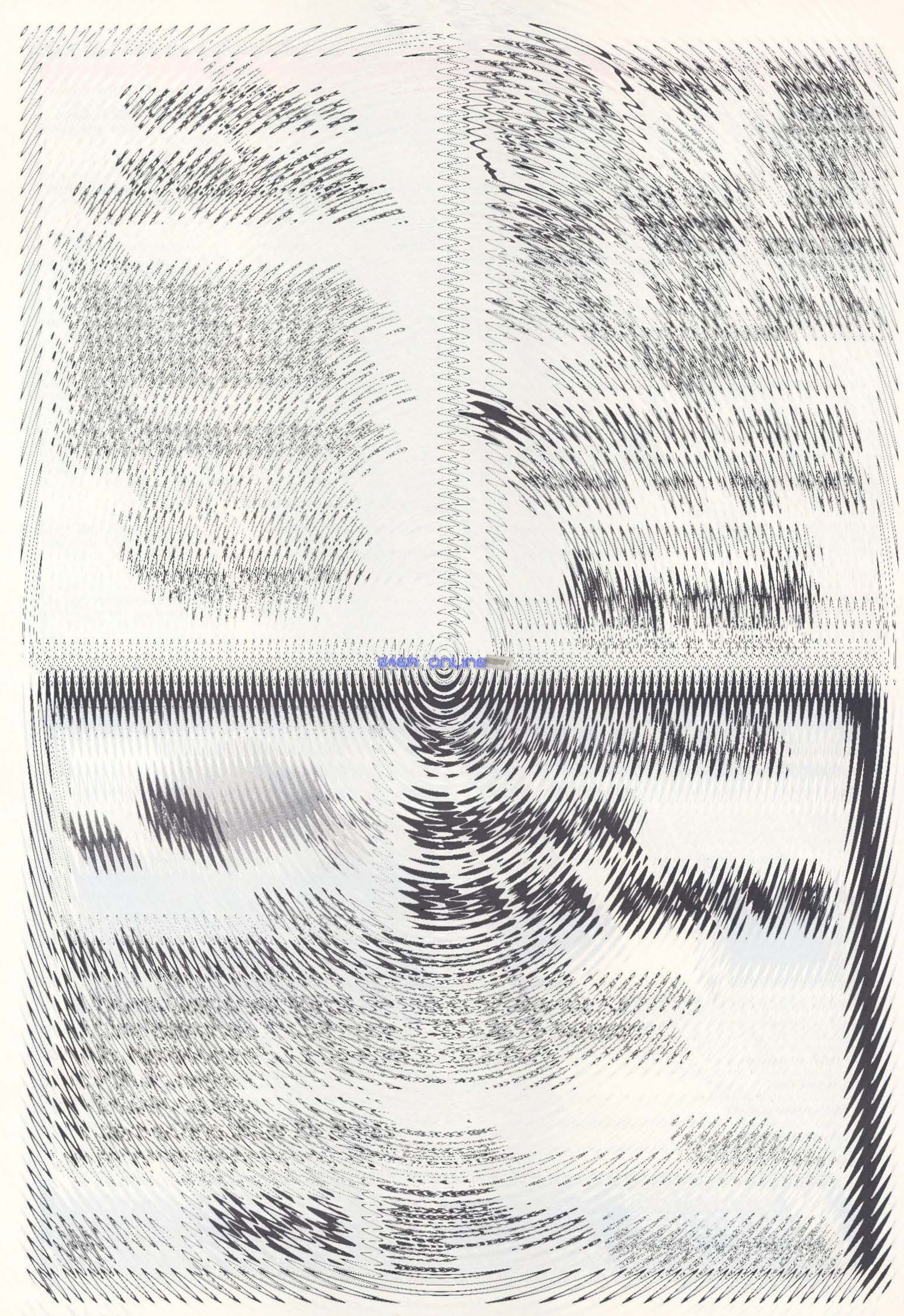


Am besten verwenden Sie dazu die vorbereitete Auftragskarte am Anfang des Heftes. Bitte beachten Sie: Ihr Anzeigentext darf maximal 5 Zeilen mit je 32 Buchstaben betragen. Überweisen Sie den Anzeigenpreis von DM 5,— auf das Postscheckkonto Nr. 14199-803 beim Postscheckamt mit dem Vermerk »Markt & Technik, 64'er«, oder schicken Sie uns DM 5,— als Scheck, in Briefmarken oder in Bargeld. Der Verlag behält sich die Veröffentlichung längerer Texte vor. Kleinanzeigen, die entsprechend gekennzeichnet sind oder deren Text auf eine gewerbliche Tätigkeit schließen läßt, werden in der Rubrik »Gewerbliche Kleinanzeigen« zum Preis von DM 10,— je Zeile Text veröffentlicht.



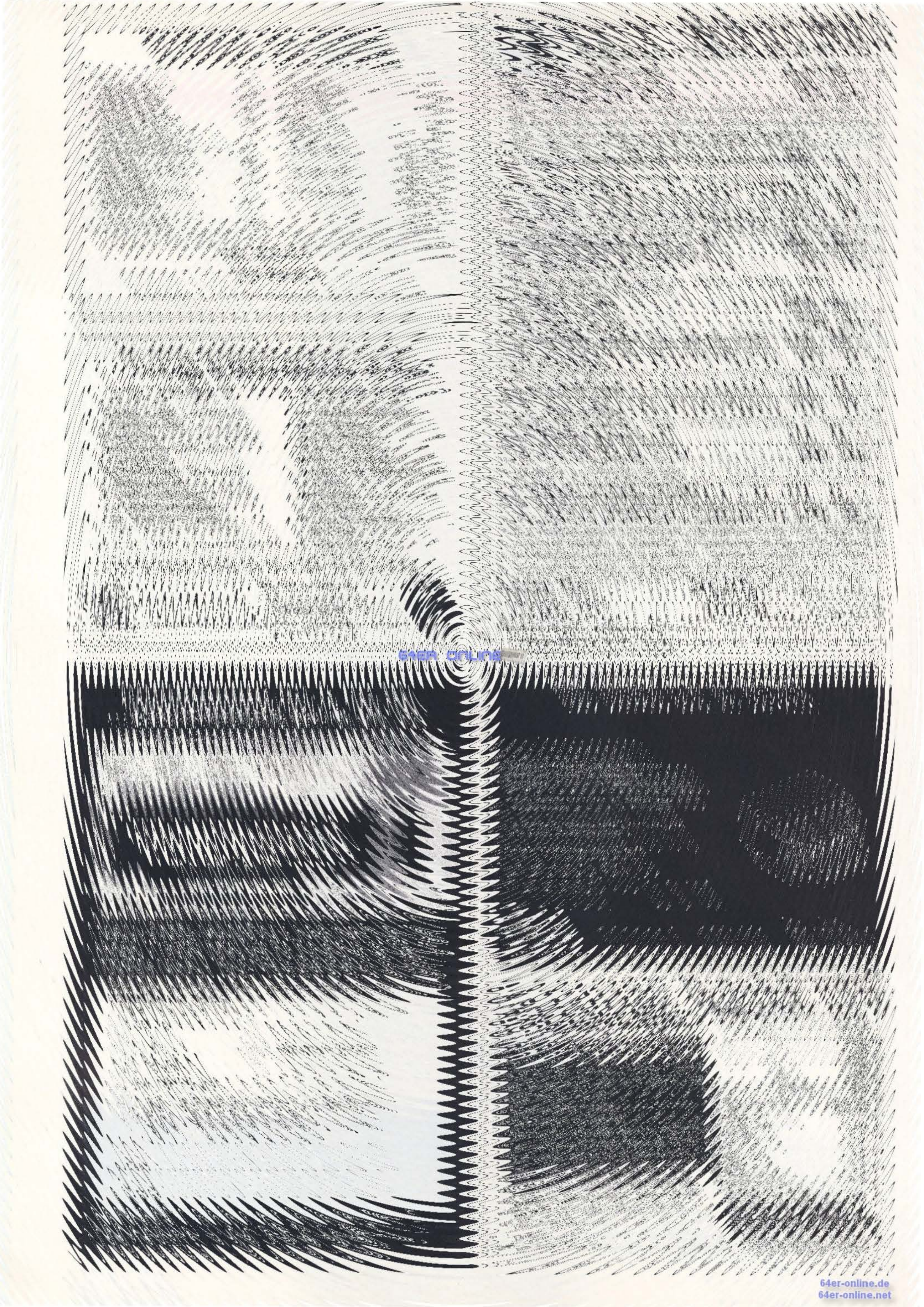


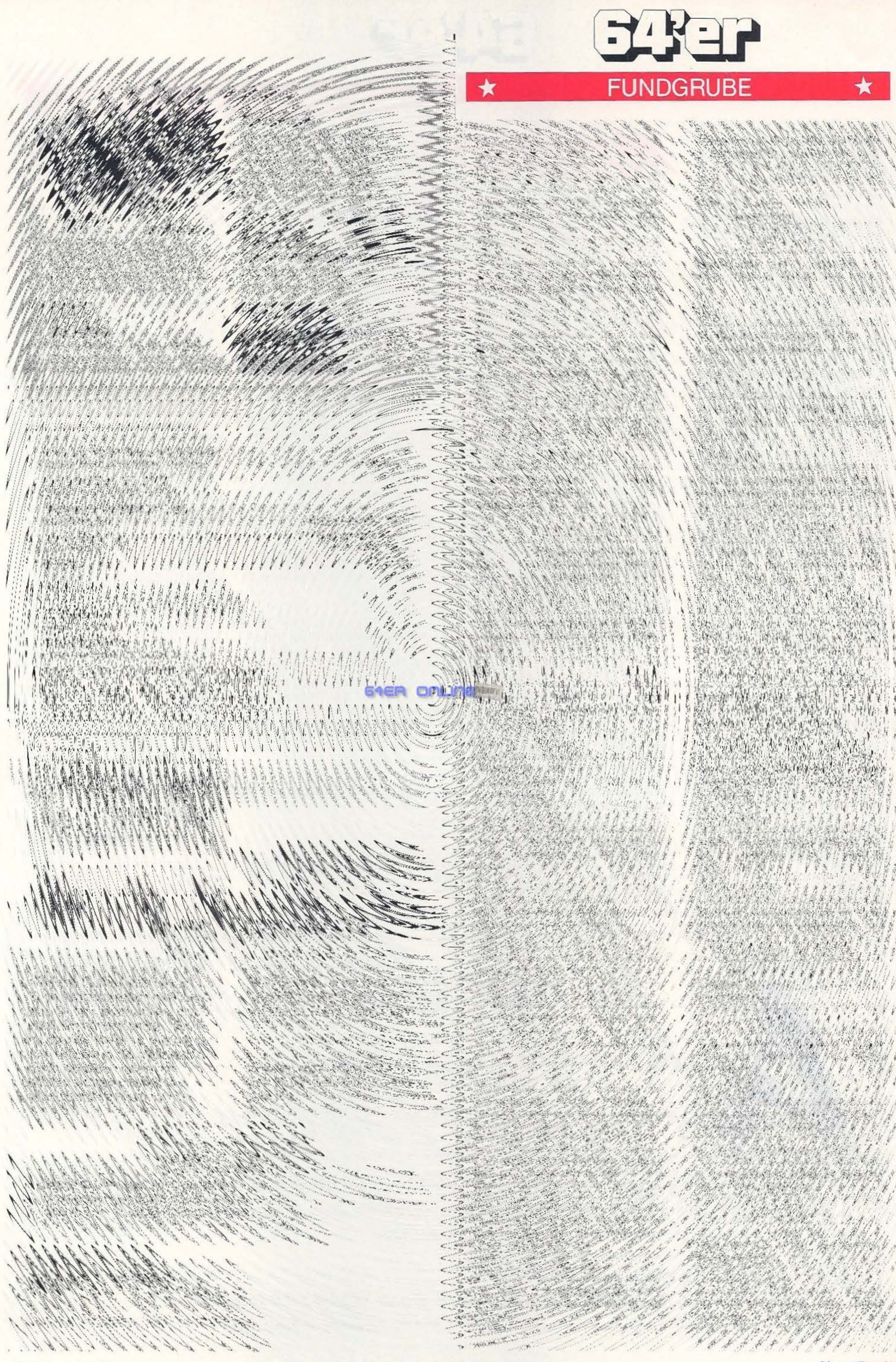


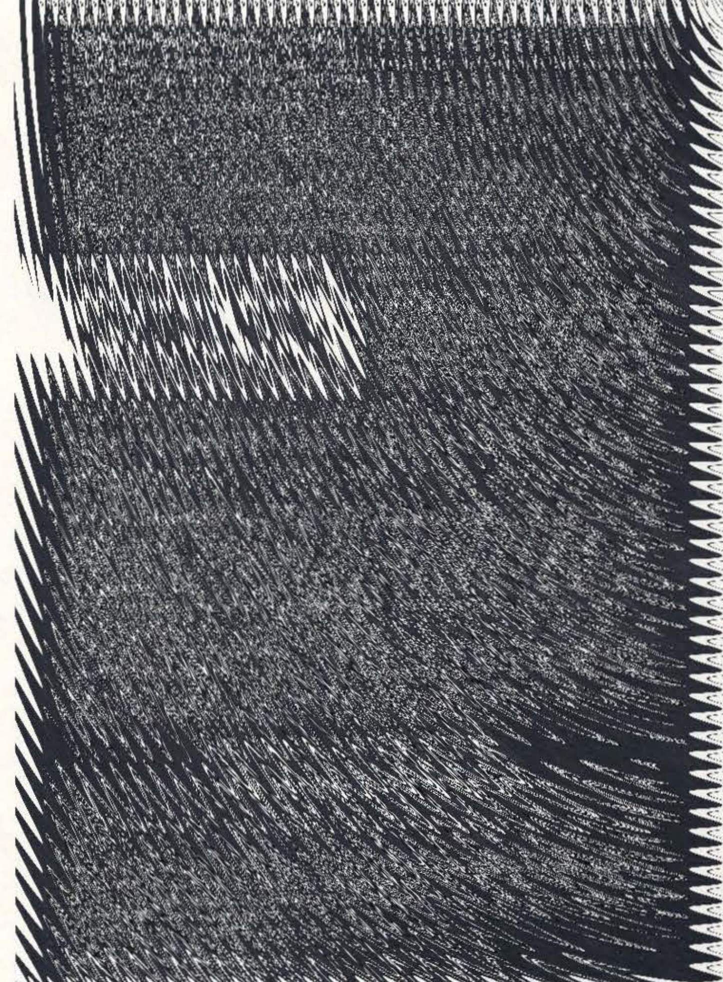

64er online

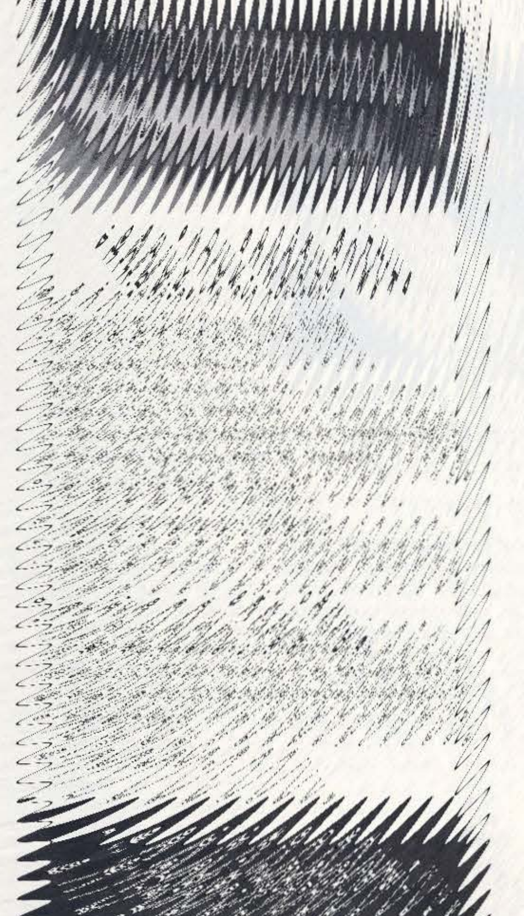






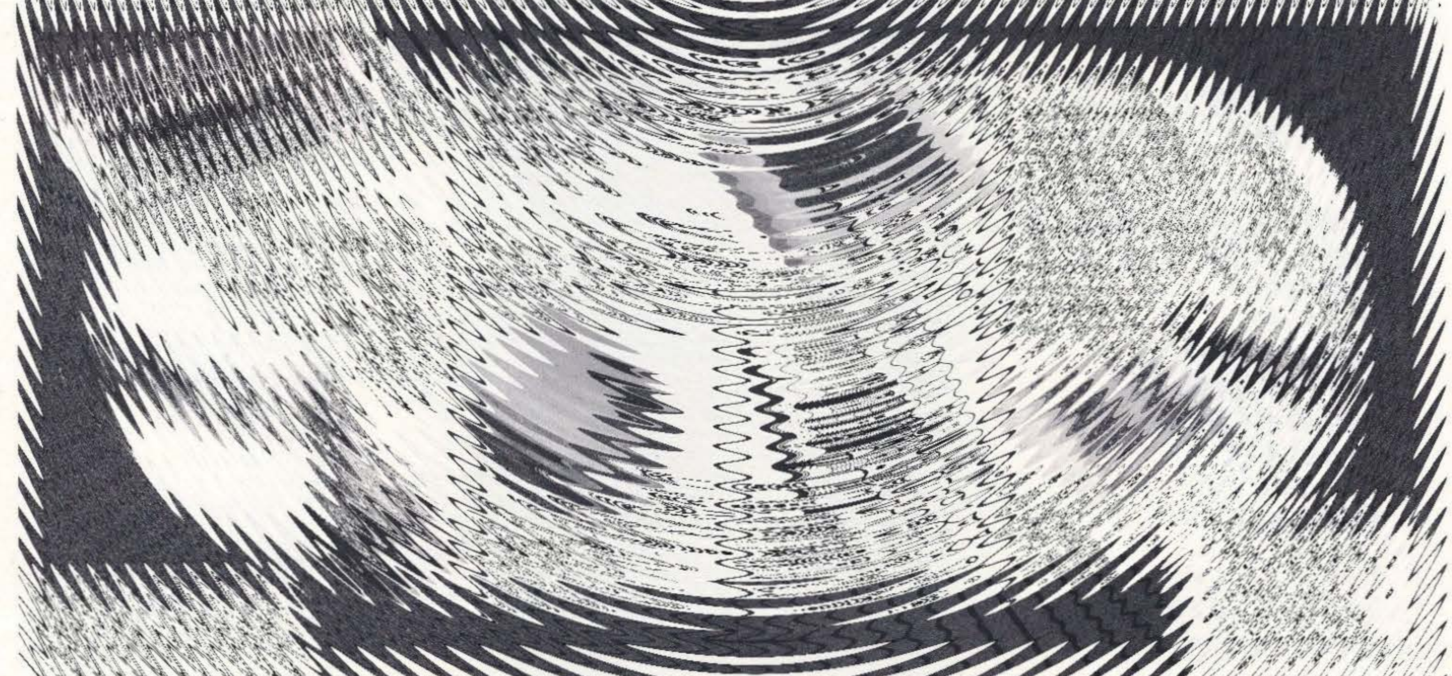




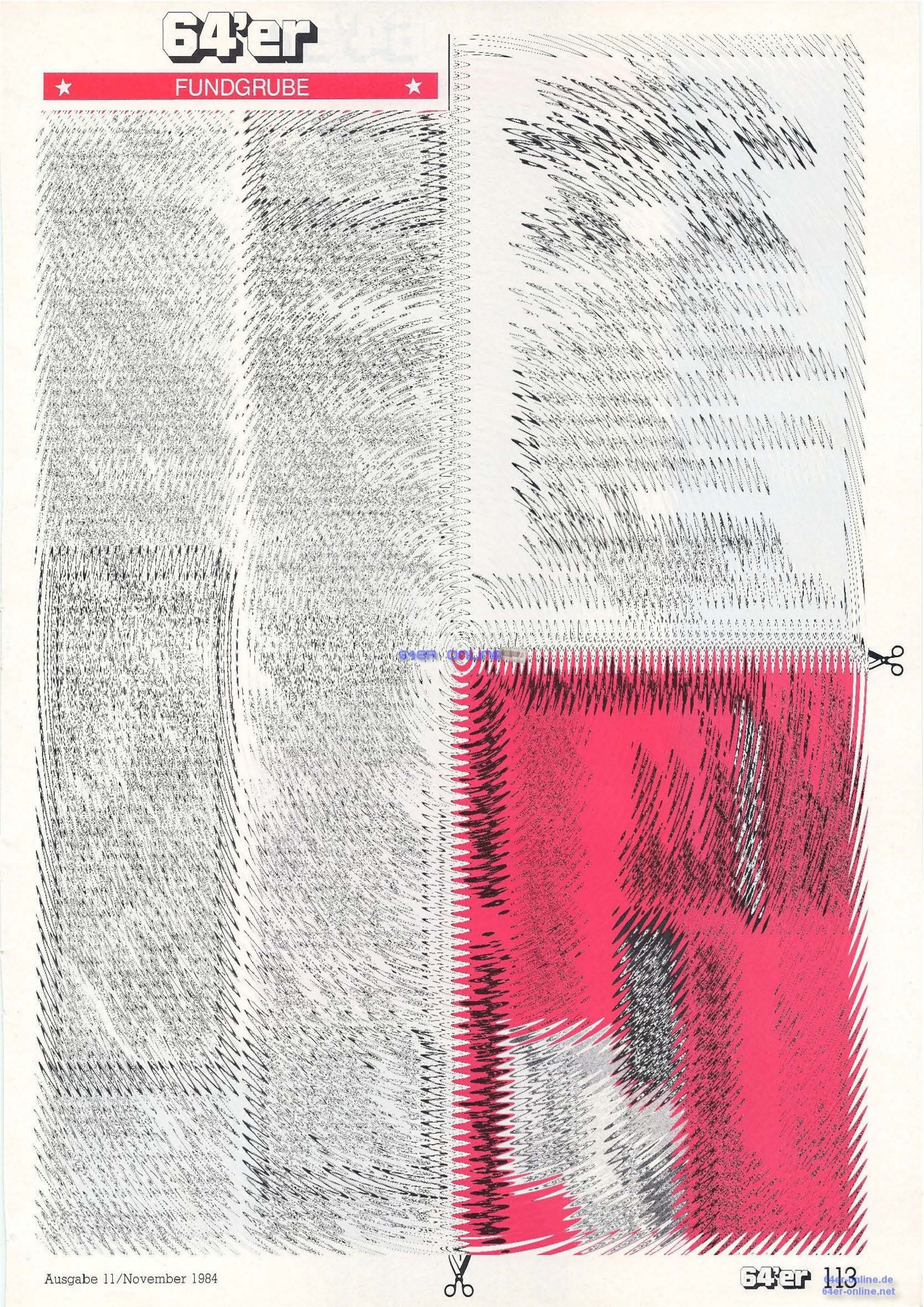




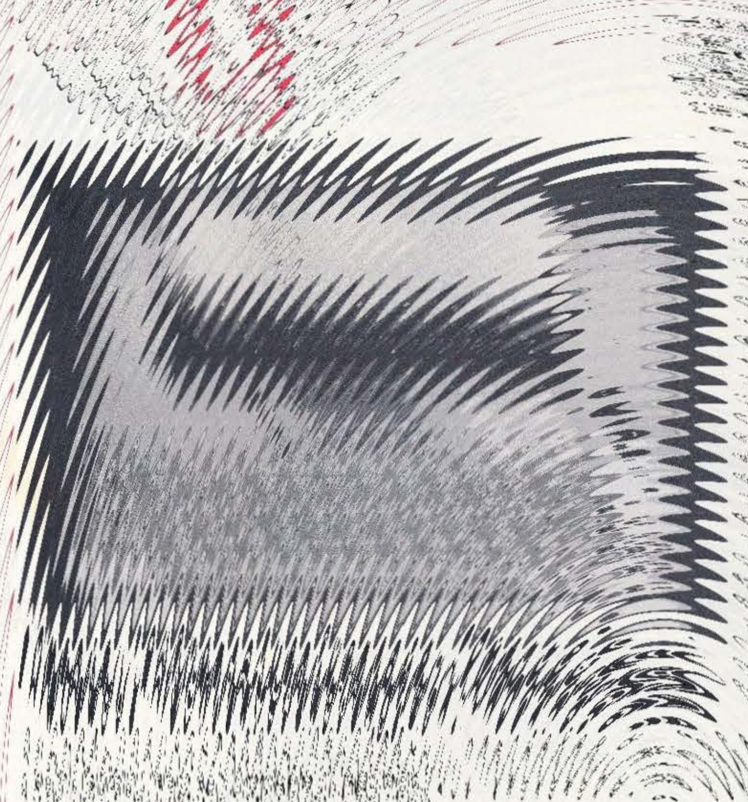
FUNDGRUBE ★ FUNDGRUBE ★ FUNDGRUBE ★ FUNDGRUBE




64'er







In die Geheimnisse der Floppy eingetaucht



Diese Folge befaßt sich mit dem Befehlssatz der VC 1541 und deren Meldungen an den Computer. Sie werden erkennen, daß Sie neben Ihrem C 64 noch einen anderen vollständigen Computer vor sich haben, der nicht nur als einfaches und »dummes« Peripheriegerät verstanden werden will.

Sicherlich machte sich mancher Floppybesitzer, der ein schnelleres Peripheriegerät als die Datasette haben wollte, schon seine Gedanken über den Preis der VC 1541: »Die kostet ja mehr als der Computer!«. In der Tat ist die VC 1541 von dieser Seite her betrachtet nicht gerade günstig, wer sich jedoch schon intensiver mit ihr beschäftigt hat, wird eine Eigenart festgestellt haben, die sie mit allen CBM-Floppys teilt: Es handelt sich hier um sogenannte Floppystationen, nicht nur um Laufwerke. Das bedeutet, diese Geräte besitzen ein eigenes Betriebssystem (DOS) und eigene Mikroprozessoren. Sie arbeiten völlig unabhängig vom Computer und dessen Speicher. Der Vorteil liegt auf der Hand: Die Floppy beansprucht weder Speicherplatz noch Rechenzeit des Computers, außer beim direkten Datenaustausch. Als Beispiel betrachte man den Befehl »N:« (Formatieren). Während der Formatierung steht der Computer zur (fast) freien Verfügung, da dieser Vorgang nur floppyintern abläuft und sich der C 64 mit READY meldet, während die 1541 noch arbeitet.

Wir wollen uns jedoch nur den Direktzugriffsbefehlen und den Speicherbefehlen widmen; auch übergehen wir die im Commodore-Handbuch nicht erwähnte relative Datenspeicherung, über die in anderen Ausgaben schon ausführlich gesprochen wurde. Uns sollen nur die Befehle beschäftigen, die uns zur willkürlichen Manipulation von Floppystation und Disketten nützen.

Zur Beruhigung: Ein Beschädigen der 1541 durch direkte Eingriffe in das DOS ist nicht zu befürchten, auch wenn es passieren kann, daß sich die Floppy nur durch Aus-/Einschalten wieder in den Normalzustand versetzen läßt. Haben Sie übrigens einmal, wie in der letzten Folge empfohlen, das Formatkennzeichen einer Diskette verändert? Sie werden sicherlich bemerkt haben, daß sich danach nichts mehr auf Ihre Diskette schreiben läßt. Mit diesem Trick, der die gleichen Folgen wie das Anbringen einer Schreibschutzplakette an der Diskette hat, können Sie sich also ganz einfach Ihre Diskette gegen unbeabsichtigtes Löschen sichern. ACHTUNG: Diese Methode funktioniert natürlich nicht, wenn neu formatiert werden soll; hiergegen hilft nur das Anbringen einer Schreibschutzplakette!

Die Floppystation verfügt über, außer den schon bekannten Befehlen zur Diskettenorganisation, noch eine ganze Anzahl weiterer Befehle, mit denen sich ungeahnte Möglichkeiten ergeben, zum Beispiel Herstellen eines eigenen Diskettenformats, Leseschutz von Disketten, Programmschutz, Modifikation der Lade- und Saveroutinen und, und... Dafür ist es allerdings nötig, daß wir diese Befehle Schritt für Schritt kennen lernen, bevor wir auf die Tricks der Profis, die Manipulationen des DOS und den gezielten Eingriff in den Programmablauf der Floppystation zu sprechen kommen. Dafür ist allerdings das Beherrschen des C 64 und der Maschinensprache unerläßlich. So lohnt es sich unter Umständen, nachdem man aus Basic nichts mehr herausholen kann, den Einstieg in die Assemblerprogram-

Bild 1. Die Speicheraufteilung der 1541

Bild 2. So ist das 2 KByte RAM der 1541 aufgeteilt.

mierung zu wagen. Sehr gute Literatur dafür ist vorhanden. Aber diesmal wollen wir uns noch auf Basic beschränken, um Sie mit dem Befehlssatz der Floppy vertraut zu machen.

Wie schon erwähnt, handelt es sich bei der 1541 um einen vollständigen Computer, der ebenso wie Ihr C 64 RAM und ein Betriebssystem (DOS) im ROM besitzt.

Die genaue Aufteilung ist in Bild 1 zu sehen. Jetzt soll uns nur der RAM-Bereich interessieren (Bild 2). Nicht nur auf der Diskette, sondern auch im RAM werden Speicherbereiche in Abschnitte zu jeweils 256 Byte aufgeteilt. Sie heißen dann nicht mehr BLOCKS sondern PAGES (Seiten). Das RAM der 1541 umfaßt nun genau 8 PAGES, durchnumeriert von 0 bis 7, insgesamt als 2 KByte. Die Page Nr. 0 (auch Zero-Page genannt) wird hier, wie auch im C 64, vom Betriebssystem als Arbeitsspeicher

```
100 REM AENDERUNG VON ID, FORMATKENN-
101 REM ZEICHEN & LEERZEICHEN ZWISCHEN
102 REM DIESEN BEIDEN.(INSG. 5 ZEICHEN)
103 REM BSP: ALTE ID :XY 2A
104 REM                ID^  ^FORMATKENNZ.
105 REM KANN AUF      :HALLO
106 REM GEAENDERT WERDEN. DAS LEERZ.
107 REM WIRD HIER ZUM ERSTEN 'L'
108 REM WIRKT SICH NUR AUF DIRECT. AUS!
109 :
110 OPEN 15,8,15,"I":OPEN1,8,2,"#"
120 PRINT#15,"U1 2 0 18 0"
130 PRINT#15,"B-P 2 162"
140 GET#1,A$,B$,C$,D$,E$
150 PRINT A$;B$;C$;D$;E$
160 INPUT"NEU:";N$
170 PRINT#15,"B-P 2 162"
180 PRINT#1,N$;
190 PRINT#15,"U2 2 0 18 0"
200 PRINT#15,"I"
210 CLOSE 8:CLOSE 15
READY.
```

Listing 1. Änderung der ID und des Formatkennzeichens

```
1000 REM UNTERPROGRAMM 1
1001 REM LESEN EINES EINTRAGES AUS DEM
1002 REM DIRECTORY (ALLE 30 BYTES !!!)
1003 REM IN DIE VARIABLE DD$
1004 REM UEBERGABEPARAMETER:
1005 REM MM=NUMMER DES EINTRAGES DER
1006 REM    GELESEN WERDEN SOLL
1007 :
1008 :
1009 :
1010 OPEN 15,8,15,"I":OPEN8,8,8,"#"
1020 NN$="":FORI=1TO30:NN$=NN$+CHR$(0):N
EXTI
1030 XX=INT((MM-1)/8)
1040 PRINT#15,"U1 8 0 18 0"
1050 FORZZ=1TOXX+1
1060 PRINT#15,"B-P 8 0"
1070 GET#8,TT$:TT=ASC(TT$+CHR$(0))
1080 GET#8,SS$:SS=ASC(SS$+CHR$(0))
1090 IF TT=0 THEN DD$=NN$:GOTO1170
1100 PRINT#15,"U1 8 0";TT;SS
1110 NEXTZZ
1120 PP=MM-(XX*8):PP=(PP-1)*32+2
1130 PRINT#15,"B-P 8";PP
1140 FORZZ=1 TO 30:GET#8,ZZ$
1150 IFZZ$=""THENZZ$=CHR$(0)
1160 DD$=DD$+ZZ$:NEXTZZ
1170 CLOSE 8:CLOSE 15
1180 RETURN
READY.
```

Listing 2. Unterprogramm 1. Lesen eines Eintrages aus dem Directory.

benutzt und steht uns deshalb nicht zur freien Verfügung. Ähnlich verhält es sich mit den Pages 1 und 2. Die Pages 3 bis 7 stellen sogenannte Pufferspeicher dar; hier werden alle Daten, die von der Diskette gelesen beziehungsweise auf sie geschrieben werden, zwischengespeichert, da nur blockweise gelesen oder geschrieben werden kann.

Soll zum Beispiel nur ein einziges Byte auf der Diskette geändert werden, so wird erst der gesamte Block in einen der 5 Pufferspeicher gelesen, dort abgeändert und schließlich komplett wieder zurückgeschrieben. Aus diesen Gründen ist es also notwendig, daß wir uns vor einem Direktzugriff einen der Puffer reservieren, in dem dann gearbeitet wird.

Mit Hilfe des »Open«-Befehls eröffnen wir einen Direktzugriffskanal. Die Syntax lautet wie folgt:

OPEN fn, gn, kn, "#"

Hierbei bedeuten:

fn — Filenummer (1-127)
gn — Gerätenummer (norm. 8)
kn — Kanalnummer in der Floppy (2-14)

Diese Abkürzungen werden wir im folgenden immer verwenden! Ein Beispiel:

OPEN 1,8,2, "#"

Diese Anweisung eröffnet im Computer ein File mit der Nummer 1, adressiert als Gerät die Floppy (Nummer 8) und reserviert in der 1541 einen Kanal (Nummer 2), dem ein Puffer zugeordnet wird. Mit den floppyinternen Kanälen verhält es sich wie folgt: Es stehen insgesamt 16 Kanäle zur Verfügung. Hierbei sind Kanal 0 und 1 für LOAD und SAVE reserviert, Kanal 15 ist der Kommandokanal, den Sie bisher immer benutzt haben, um Befehle (zum Beispiel Formatieren) an die Floppy zu senden und die Fehlermeldungen der Floppy zu empfangen.

Für unsere Zwecke stehen also noch die Kanäle 2 bis 14 zur Verfügung. In unserem Fall reserviert die Floppy den nächsten freien Puffer. Will man jedoch einen bestimmten Puffer reservieren, etwa um dort ein Maschinenprogramm abzulegen, so ist es notwendig, der 1541 mitzuteilen, welcher Puffer gewünscht wird: OPEN1,8,2," #1"

Es ist hier allerdings zu beachten, daß der gewählte Puffer nicht schon belegt ist; in diesem Fall gibt die Floppy eine Fehlermeldung aus. Wollen Sie an dieser Stelle mehr über das Auslesen der Fehlermeldungen und deren Bedeutung wissen, können wir Sie hier beruhigt auf das Commodore-Handbuch verwei-

```
2000 REM UNTERPROGRAMM 2
2001 REM SCHREIBEN EINES EINTRAGES IN
2002 REM DAS DIRECTORY (30 BYTES !!!)
2003 REM UEBERGABEPARAMETER:
2004 REM MM=NUMMER DES EINTRAGES DER
2005 REM    GESCHRIEBEN WERDEN SOLL
2006 REM DD$=DIRECTORYEINTRAG
2007 :
2008 :
2009 :
2010 OPEN 15,8,15,"I":OPEN8,8,8,"#"
2020 XX=INT((MM-1)/8)
2030 PRINT#15,"U1 8 0 18 0"
2040 FORZZ=1TOXX+1
2050 PRINT#15,"B-P 8 0"
2060 GET#8,T$:TT=ASC(T$+CHR$(0))
2070 GET#8,S$:SS=ASC(S$+CHR$(0))
2080 IF TT=0 THEN 2150
2090 PRINT#15,"U1 8 0";TT;SS
2100 NEXTZZ
2110 PP=MM-(XX*8):PP=(PP-1)*32+2
2120 PRINT#15,"B-P 8";PP
2130 PRINT#8,DD$;
2140 PRINT#15,"U2 8 0";TT;SS
2150 CLOSE 8:CLOSE 15
2160 RETURN
READY.
```

Listing 3. Unterprogramm 2. Schreiben eines Directory-Eintrages

```
100 REM BEISPIEL FUER EINE KLEINE
101 REM DIRECTORY-MANIPULATION:
102 REM SCRATCH-SCHUTZ EINZELNER FILES
103 REM NACH ANZEIGE DES FILENAMENS:
104 REM J = SCHUETZE DIES FILE
105 REM N = WEITER ZUM NAECHSTEN FILE
106 REM E = ENDE
107 REM ACHTUNG !!! "SCHUETZT" AUCH
108 REM SCHON GESCRATCHTE FILES WENN
109 REM VERLANGT, STELLT SIE ABER NICHT
110 REM WIEDER HER !!!
111 REM SCRATCH-SCHUTZ WIRD IM DIRECT.
112 REM DURCH EIN '<' HINTER DEM
113 REM FILETYP ANGEZEIGT. NAEHERES
114 REM SIEHE TABELLE FOLGE 1 !!!
115 REM ACHTUNG !!! NUR ZUSAMMEN MIT
116 REM DEN UNTERPROGRAMMEN 1 & 2
117 REM LAUFFAEHIG !!!
118 :
119 :
120 MM=0
130 MM=MM+1:DD$="":GOSUB1000
140 IF DD$=NN$THENEND
150 PRINTMID$(DD$,4,16):INPUTAA$
160 IF AA$="E"THEN END
170 IF AA$="N"THEN 130
180 HH$=LEFT$(DD$,1)
190 HH$=CHR$(ASC(HH$)OR2^6)
200 DD$=HH$+RIGHT$(DD$,29)
210 GOSUB2000
220 GOTO 130
230 END
READY.
```

Listing 4. So kann man Files schützen.

sen. Im allgemeinen sind Puffer 4 für die BAM und Puffer 3 für das Directory reserviert. Haben Sie die Wahl des Puffers der Floppy überlassen, so erfahren Sie die gewählte Nummer durch Auslesen des soeben geöffneten Direktzugriffskanals:

10 OPEN 1,8,2," #"
20 GET #1,D$
30 D = ASC (D$ + CHR$(0))
40 REM Puffernummer in D

Die BLOCK-Befehle

a) Der BLOCK-READ-Befehl (B-R): Mit dem BLOCK-READ-Befehl liest man jeden beliebigen Block von Diskette in einen vorher reservierten Puffer. Die Syntax lautet:

PRINT #fn,"B-R";kn;dn;t;s

dn — Drivenummer (immer 0)
t — Tracknummer
s — Sektornummer



```
100 REM SCHREIBSCHUTZ SETZEN
101 REM DURCH AENDERUNG DES FORMAT-
102 REM KENNZEICHENS IN DER BAM !!!
103 REM FUNKTIONSWEISE :
104 REM FORMATKENNZEICHEN WIRD AUF
105 REM BELIEBIGEN WERT AUSSER 'A'
106 REM GESETZT. AB SOFORT KOENNEN
107 REM KEINE SCHREIBVORGAENGE AUSSER
108 REM FORMATIEREN DURCHGEFUEHRT
109 REM WERDEN. ALSO VORSICHT !
110 :
200 OPEN 15,8,15,"I":OPEN8,8,8,"#"
210 PRINT#15,"U1 8 0 18 0"
220 PRINT#15,"B-P 8 2"
230 PRINT#8,"X";
240 PRINT#15,"U2 8 0 18 0"
250 PRINT#15,"I"
260 CLOSE8:CLOSE15:END
READY.
```

Listing 5. Schützen Sie Ihre Diskette vor jedem Schreibzugriff.

Beispiel: PRINT#15,"B-R 2 0 18 0"

Diese Befehlsfolge liest den Block 18,0 von der Diskette in den oben reservierten Puffer. Wie man sieht, können anstelle der CHR$-Codes feste Zahlenwerte in den Befehlsstring mit übernommen werden. Das ganze hat bloß einen kleinen Schönheitsfehler. Mit dem B-R-Befehl läßt sich das erste Byte eines Blocks nicht lesen. Deshalb benutzt man normalerweise anstatt des B-R-Befehls den U1-Befehl. Dieser hat exakt die gleiche Syntax und kann in jedem Fall benutzt werden:

PRINT#15,"U1 2 0 18 0"

Auf diese USER-Befehle kommen wir später zurück. Mit einer GET#-Schleife lassen sich nun die einzelnen Bytes in den Computer einlesen.

b) Der BLOCK-WRITE-Befehl (B-W):
Hiermit lassen sich die Daten aus dem reservierten Puffer wieder auf die Diskette schreiben. Syntax:
PRINT#fn,"B-W";kn;dn;t;s
Beispiel: PRINT#15,"B-W 2 0 18 0"
Natürlich gibt es analog zum B-R einen USER-Befehl ; U2.
Beispiel: PRINT#15,"U2 2 0 18 0".

c) Der BUFFER-POINTER-Befehl (B-P):
Für jeden Puffer gibt es einen Zeiger, den BUFFER-POINTER. Dieser zeigt auf das aktuelle Byte im Puffer und wird bei jedem Datenzugriff um Eins erhöht, damit man alle 256 Bytes eines Blocks der Reihe nach lesen kann. Dieser Pointer wird mit dem B-P-Befehl gezielt auf bestimmte Bytes positioniert, wenn man nur einzelne Werte und nicht den gesamten Block lesen will. Syntax:
PRINT#fn,"B-P";kn;position
Beispiel:
Wir möchten in die Variable A den Wert des 123. Bytes von Block 1;16 einlesen:
10 OPEN15,8,15
20 OPEN1,8,2,"#"
30 PRINT#15,"U1 2 0 1 16"
40 PRINT#15,"B-P 2 122"
50 GET#1,A$
60 A=ASC (A$+CHR$(0))
Als weiteres Beispiel dient Listing 1.

d) Der BLOCK-ALLOCATE-Befehl (B-A):
Wenn Sie im Direktzugriffsverfahren eine Diskette beschreiben, muß in der BAM danach auch verzeichnet werden, daß die entsprechenden Blocks mit Daten gefüllt sind und nicht mehr überschrieben werden dürfen. Dazu dient der B-A-Befehl, der jeden beliebigen Block in der BAM als belegt kennzeichnet. Die Syntax lautet:
PRINT#fn,"B-A";dn;t;s
Beispiel:
PRINT#15,"B-A 0 1 16"
kennzeichnet Block 1;16 als belegt; war dieser Block schon belegt, meldet sich die Floppy mit der Fehlermeldung »65,NO BLOCK,XX,YY«; wobei XX und YY die Track- und Sektornummer des nächsten freien Blocks angeben.

e) Der BLOCK-FREE-Befehl (B-F):
Dieser ist das genaue Gegenstück zum B-A-Befehl; er deklariert einmal belegte Blöcke wieder als frei für einen weiteren Zugriff. Seine Syntax ist identisch mit der des B-A-Befehls.


```
100 REM SCHREIBSCHUTZ SETZEN / LOESCHEN
101 REM DURCH AENDERUNG DES FORMAT-
102 REM KENNZEICHENS IN DER BAM !!!
103 REM FUNKTIONSWEISE SETZEN :
104 REM FORMATKENNZEICHEN WIRD AUF
105 REM BELIEBIGEN WERT AUSSER 'A'
106 REM GESETZT. AB SOFORT KOENNEN
107 REM KEINE SCHREIBVORGAENGE AUSSER
108 REM FORMATIEREN DURCHGEFUEHRT
109 REM WERDEN. ALSO VORSICHT !
110 REM FUNKTIONSWEISE FREIGEBEN :
111 REM DIE FLOPPY SPEICHERT DAS
112 REM FORMATKENNZEICHEN DER EINGEL.
113 REM DISKETTE IN DER SPEICHERSTELLE
114 REM $0101 ZWISCHEN. WIRD DIES
115 REM VOR DEM SCHREIBVORGANG AUF
116 REM 'A' ZURUECKGESETZT, LAESST
117 REM SICH DIE FLOPPY 'UEBERLISTEN'
118 REM DAS 'A' WIRD NUN IN DIE BAM
119 REM GESCHRIEBEN
120 :
130 PRINT"WOLLEN SIE DIE EINGEL. DISKETT
E"
140 INPUT"SCHUETZEN ODER FREIGEBEN":AA$
150 IF AA$="S"THEN 200
160 IF AA$="F"THEN 300
170 PRINT:RUN
180 :
190 :
200 OPEN 15,8,15,"I":OPEN8,8,8,"#"
210 PRINT#15,"U1 8 0 18 0"
220 PRINT#15,"B-P 8 2"
230 PRINT#8,"X";
240 PRINT#15,"U2 8 0 18 0"
250 PRINT#15,"I"
260 CLOSE8:CLOSE15:PRINT:RUN
270 :
280 :
290 :
300 OPEN 15,8,15,"I":OPEN8,8,8,"#"
310 PRINT#15,"U1 8 0 18 0"
320 PRINT#15,"B-P 8 2"
330 PRINT#15,"M-W"CHR$(1)CHR$(1)CHR$(1)C
HR$(65)
340 PRINT#8,"A";
350 PRINT#15,"U2 8 0 18 0"
360 PRINT#15,"I"
370 CLOSE8:CLOSE15:PRINT:RUN

READY.
```

Listing 6. Eine einfache Routine, um das Directory zu sortieren.

```
100 REM BEISPIEL FUER MEMORY-EXECUTE
101 REM LOEST IN DER FLOPPY LANGSAMES
102 REM BLINKEN DER ROTEN LED (KENN-
103 REM ZEICHNET NORMALERWEISE HARD-
104 REM WARE-FEHLER) AUS.
105 REM KANN NUR DURCH AUSLOESEN EINES
106 REM RESETS ENTWEDER DURCH EIN/AUS-
107 REM SCHALTEN DER FLOPPY ODER DES
108 REM COMPUTERS BEENDET WERDEN.
109 REM EINSPRUNGSADRESSE : $EA6E
110 :
120 OPEN15,8,15
130 PRINT#15,"M-E"CHR$(110)CHR$(234)
140 CLOSE 15
150 END
READY.
```

Listing 7. Simulieren Sie einen Hardware-Fehler

f) Der BLOCK-EXECUTE-Befehl (B-E): Dieser Befehl nimmt eine Sonderstellung ein. Er gleicht im Prinzip dem B-R-Befehl; nur mit dem zusätzlichen Effekt, daß der eingelesene Block im Puffer als Maschinenprogramm gestartet wird.

Zur Vertiefung der Block-Befehle sei noch auf die Listings 2 bis 5 hingewiesen, welche die eben besprochenen Anwendungen noch an praktischen Beispielen verdeutlichen.

Die MEMORY-Befehle

a) Der MEMORY-READ-Befehl (M-R): Dieser Befehl entspricht haargenau dem PEEK-Befehl in Basic. Mit ihm können Sie jede beliebige Speicherstelle der Floppy auslesen.
Syntax:
PRINT#fn,"M-R";CHR$(adl);CHR$(adh);CHR$(n)
adl = Low-Byte
adh = High-Byte
n = Anzahl (0 bis 255)

Abgeholt werden die gelesenen Daten ebenfalls über den Kommandokanal mit GET#.
Beispiel: Lesen der beiden ID-Zeichen im ASCII-Format der zuletzt initialisierten Diskette:
10 OPEN15,8,15
20 PRINT#15,"M-R"CHR$(18)CHR$(0)CHR$(2)
30 GET#15,A$,B$
40 PRINTA$;B$

Diese Routine liest die Zero-Page Adressen 18 und 19, in denen die entsprechenden Werte gespeichert sind. In Tabelle 1 sind einige der wichtigsten Zero-Page Adressen aufgeführt.

b) Der MEMORY-WRITE-Befehl (M-W):
Dieses Kommando kann als POKE-Befehl in den Floppy-Speicher angesehen werden. Die Syntax ist hier wie folgt:
PRINT#fn,"M-W";CHR$(adl)CHR$(adh)CHR$(n)CHR$(data1)CHR$(data2)...

c) Der MEMORY-EXECUTE-Befehl (M-E):

Fortsetzung auf Seite 163

Assembler ist keine Alchimie

Teil 3

In den ersten beiden Folgen unseres Assembler-Kurses mußten Sie noch mit Basic-Ladern arbeiten. Jetzt steht Ihnen ein leistungsfähiger Monitor zur Verfügung, der SMON. Somit können Sie alle Beispiele direkt eingeben und ausprobieren.

In der letzten Folge haben wir die ersten Assembler-Befehle kennengelernt und wissen, wie man sie benutzt und was sich im Computer dabei tut. Die Zahlen der Assembler-Alchimisten haben uns einige Geheimnisse enthüllt, obwohl sie für die Zweifingerlinge und die Sechzehnfingerlinge gedacht sind. Die Binärzahlen können wir schon zusammenzählen. Heute werden Sie eine Reihe weiterer Assembler-Befehle kennenlernen und noch ein weiteres Zahlensystem. Wir ergründen das Geheimnis der negativen Zahlen und machen uns die Funktion der Flaggen zunutze.

0800	00	0C	08	0A	00	41	25	B2
		080C Koppeladresse		000A Zeilennr.10		A	%	= Token
0808	AB	31	32	00	12	08	14	00
	— Token	1	2	Zeilenende	0812 Koppeladresse		0014 Zeilennr.20	
0810	80	00	00	00	FF	FF	FF	FF
	END Token	Zeilenende	Programmende		Leerer Speicher			

Bild 1. Der Monitor zeigt das nackte Programm im Speicher

Wir haben nun auch einen sehr brauchbaren Assembler für den C 64: Den SMON, dessen 1. Teil in dieser Ausgabe abgedruckt ist. Künftig wird in dieser Serie die SMON-Syntax verwendet und kein Basic-Lader mehr angegeben. Außerdem hat in Ausgabe 9 die Serie »Der gläserne VC 20« begonnen, so daß sich der Schwerpunkt hier mehr auf den C 64 verlagert. Das sollte aber die VC 20-Fans nicht davon abhalten, diesen Kurs weiter zu verfolgen, denn bis auf gelegentliche Adreßänderungen ist fast alles für sie verwendbar.

Eine Zauberformel der Assembler-Alchimisten: INX, INY, INC, DEX, DEY, DEC?

Wir wissen ja schon, daß man diese »Zauberformeln« entzaubern kann. INX heißt einfach »increment X-Register«, also Inhalt des X-Registers um 1 erhöhen. Es wird Ihnen sicher einleuchten, daß INY dasselbe mit dem Y-Register tut. Etwas weniger deutlich ist das bei INC. Das bedeutet »increment memory«, also zähle zum Inhalt einer Speicherstelle eins dazu. INX und INY enthalten alles, was dem Computer zu sagen ist, sind also offensichtlich 1-Byte-Befehle mit der in der letzten Folge schon kennengelernten impliziten Adressierung. Bei INC muß dem Computer noch gesagt werden, welche Speicherstelle er um 1 erhöhen soll. Es gehört also noch eine Adresse dazu. Das läßt diesen Befehl im allgemeinen zu einem 3-Byte-Befehl werden.

Befehle zum Zählen

Das umgekehrte leisten die Befehle DEX, DEY und DEC. Sie bedeuten nämlich »decrement X-Register«, also »zähle das X-Register um eins herunter«, beziehungsweise das Y-Register oder — bei DEC — die angegebene Speicherstelle. Für die Adressierungsart und die Anzahl Bytes pro Befehl gilt hier das gleiche wie für die INX...-Befehle. Sehen wir uns das an einem kleinen Beispiel an: Bitte lesen Sie sich dazu die Bedienungshinweise zum SMON durch.

```
1500   LDA #00
1502   LDX #01
1504   STA D800
1507   STX 0400
150A   INX
150B   STA D801
150E   STX 0401
1511   DEX
1512   STA D802
1515   STX 0402
1518   BRK
```

Wenn Sie das kleine Programm mit G 1500 starten, dann sollten Sie in der linken oberen Ecke des Bildschirms ABA in schwarzer Schrift stehen haben. Was ist geschehen? Wir haben den Inhalt des Akku (= 0, also Farbcode für schwarz) in das Bildschirm-Farbregister geschrieben (#D800), dann den Inhalt des X-Registers (1 = POKE-Code für den Buchstaben A) in die erste Bildschirm-Speicherzelle (#0400). Anschließend wurde das X-Register um 1 erhöht (2 = POKE-Code für den Buchstaben B) und dieser Inhalt in die zweite Bildschirmzelle geschrieben. Außerdem mußte natürlich auch dieser Bildschirm-Farbspeicherplatz mit dem Farbcode 0 belegt werden. Durch DEX wurde das X-Register wieder herunter gezählt, somit wieder ein A erzeugt und in die dritte Bildschirmstelle gedruckt.

Sie haben sicher schon bemerkt, daß man auf diese Weise Abläufe mitzählen kann. Soll zum Beispiel ein Vorgang 20 mal wiederholt werden, dann packt man ins X-Register (oder ins Y-Register oder in eine andere Speicherstelle) den Anfangswert 0, läßt den Computer eine Arbeit ausführen, erhöht das entsprechende Register oder die Speicherzelle um 1 mit INX, INY oder INC, prüft dann, ob dieser Inhalt schon 20 geworden ist und so weiter. Wie man diese Prüfung vornimmt, dazu kommen wir erst später bei den BRANCH-Befehlen. Das ist also ähnlich wie im Basic bei den FOR...NEXT-Schleifen: Dort

Assembler ist keine Alchimie

wird eine Variable als Zähler verwendet, hier ein Register (oder eine Speicherstelle). Ebenso wie im Basic bei diesen Schleifen kann man auch hier rückwärts zählen mit DEX, DEY oder DEC. Das hat oft gewisse Vorzüge, was uns aber noch nicht kümmern soll.

Wenn wir diese Befehle als Zähler verwenden, sollten wir im Auge behalten, daß eine Speicherstelle (auch ein X- oder Y-Register) Zahlen nur von 0 bis 255 enthalten kann. Die höchste 8-Bit-Zahl ist ja:

```
dez. 255 = bin. 1111 1111
     +1             1
          ————————————— +
ergibt:     (1) 0000 0000
```

Wenn wir also über 255 hinauszählen, ergibt sich wieder 0 und so weiter, weil ein Überlauf stattgefunden hat. Das 9.Bit paßt nicht mehr in das Byte hinein. Um nochmal genau sehen zu können, was unser Computer da tut, probieren Sie einmal aus:

```
1500 LDA #01
1502 BRK
```

Das soll uns die Register zunächst mal im Ausganszustand zeigen. Nach G 1500 werden sie angezeigt:

```
AC XR YR N V- BDI ZC
01 00 00 0 0 110 000
```

Im Akku steht jetzt die dort eingeladene 1. Nun wollen wir das X-Register laden mit 255 (also $FF). Dazu ändern wir das Programm:

```
1502 LDX #FF
1504 BRK
```

Nach erneutem G 1500 zeigen die Register:

```
AC XR YR N V- BDI ZC
01 FF 00 1 0 110 000
```

Im X-Register steht nun die Zahl $FF. Bei den Flaggen hat sich die N-Flagge (die negative Zahlen anzeigen soll) auf 1 geschaltet!

Nun wollen wir das X-Register über 255 hinauszählen. Wir verändern das Programm nochmal:

```
1504 INX
1505 BRK
```

Der Start mit G 1500 liefert uns die folgende Registeranzeige:

```
AC XR YR N V- BDI ZC
01 00 00 0 0 110 010
```

Wie erwartet, ist der Überlauf des X-Registers eingetreten: Es ist jetzt Null. Die N-Flagge hat ihren gewohnten Wert 0 wieder angenommen und die Z-Flagge, die uns anzeigt, ob die letzte Operation eine Null erzeugt hat, ist jetzt gesetzt. Bei weiterem Hochzählen verschwindet die Z-Flagge wieder:

```
1505 INX
1506 BRK
```

G 1500 liefert den Registerinhalt:

```
AC XR YR N V- BDI ZC
01 01 00 0 0 110 000
```

Das gleiche passiert bei Verwendung des Y-Registers als Zähler, wie Sie leicht durch Austauschen aller auf X bezogenen Befehle feststellen können. Sehr nett ist es, diesen Befehlsablauf einmal für den INC-Befehl auf die Speicherstelle $0400 (Bildschirmspeicher links oben) bezogen ablaufen zu lassen. Wenn man darauf achtet, daß kein Hochscrollen des Bildschirms eintritt, kann man das Ergebnis außer in den Registern auch noch als Zeichen auf dem Bildschirm verfolgen. Der Beginn der Befehlsequenz ist dann sinnvollerweise:

```
1500 LDA #FF
1502 STA 0400
1505 BRK
```

Im folgenden setzt man dann anstelle von INX immer INC 0400 ein.

Was passiert beim Herunterzählen unter Null? Sie können das mit der gezeigten Befehlskette leicht verfolgen, indem Sie immer statt INX jetzt DEX setzen und die Register nicht mit $FF, sondern mit 01 laden. Es zeigt sich, daß beim Herabzählen nach der Null wieder 255 (= $FF) im Register zu finden ist. Die Reaktion der N- und der Z-Flagge auf den jeweiligen Registerinhalt ist die gleiche wie beim Hochzählen.

Es ist uns nun deutlich, daß diese sechs Befehle die N-Flagge und die Z-Flagge beeinflussen können. Diese Tatsache wird später noch eine große Rolle spielen, wenn es um die bereits erwähnte Schleifenkontrolle geht.

Noch ein alchimistischer Zahlentrick

Die Assembler-Alchimisten haben noch viel mehr Arten der Zahlen- und Zeichendarstellung auf Lager. Eine davon ist die Codierung als BCD-Zahlen. BCD kommt vom englischen »binary coded dezimal«, was bedeutet: Binär codierte Dezimalzahlen.

Zwischendurch möchte ich noch eine Bemerkung loswerden, die Sie als Trost auffassen sollen: Auch wenn wir später andere Zahlendarstellungen kennenlernen werden, es wird nicht so schwierig! Sogar so komplette Idioten wie Computer verstehen das, obwohl man ihnen alles haarklein vorkauen muß.

Speicherstelle	0814	0815
Byte	1	2
Inhalt	C1 1100 0001	80 1000 0000
	Kennbits 7 für Integer	
	0100 0001 ≙ 65 Code für A	0000 0000

Bild 2. So werden Integer-Variable aus Basic-Program

Wenden wir uns nun wieder den lächerlich einfachen BCD-Zahlen zu. Alle Zahlen von 0 bis 9 lassen sich binär mit nur 4 Bits ausdrücken:

Binär	Dezimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Die weiteren Werte 1010 bis 1111 werden in der BCD-Codierung nicht benutzt. Liegt nun eine Dezimalzahl (zum Beispiel 12) vor, dann wird jede Stelle dieser Zahl (also die 1 und die 2) getrennt binär codiert. In unserem Beispiel mit der 12 wäre das dann 0001 für die 1 und 0010 für die 2. Somit ist die 12 im BCD-Code 0001 0010. Jede Ziffer erhält so ihr Nibble. Eine Zahl im BCD-Format hat deswegen keine feste Anzahl von Bytes, sondern die Byte-Zahl hängt von der Anzahl der Stellen ab. Die Zahl 1984 beispielsweise braucht 2 Bytes: 0001 1001 1000 0100.

Schwierig gestaltet sich das Rechnen mit diesen Zahlen wegen der sechs unbenutzten Codes. Aber auch da habe ich einen Trost für Sie: Wir werden damit nicht rechnen. Wozu das ganze dann, werden Sie sich fragen? Der Grund für das alles ist, daß BCD-Zahlen im Gegensatz

Teil 3

zu den Zahlen mit festem Format (die sonst verwendet werden) so eingegeben und verarbeitet werden können, wie sie vorliegen. Das ist im kaufmännischen Bereich manchmal notwendig, wo eben 1000mal 0,1 Pfennige 1 Mark ergeben und Fehler unzulässig sind. Sollten Sie also vor dem Problem stehen, mit BCD-Zahlen rechnen zu müssen, grämen Sie sich nicht: Unser Prozessor kennt den Dezimalmodus. Er ist dann eingeschaltet, wenn die Dezimal-Flagge auf 1 gesetzt ist.

0816	0817	0818 bis 081A
3	4	5 — 7
FF	F4	00 00 00
111 1111	1111 0100	unbenutzt bei Integerzahlen
MSB	LSB	
von —12		
Variablenname und -typ Variablenwert		

nen vom C 64 im Speicher eingerichtet

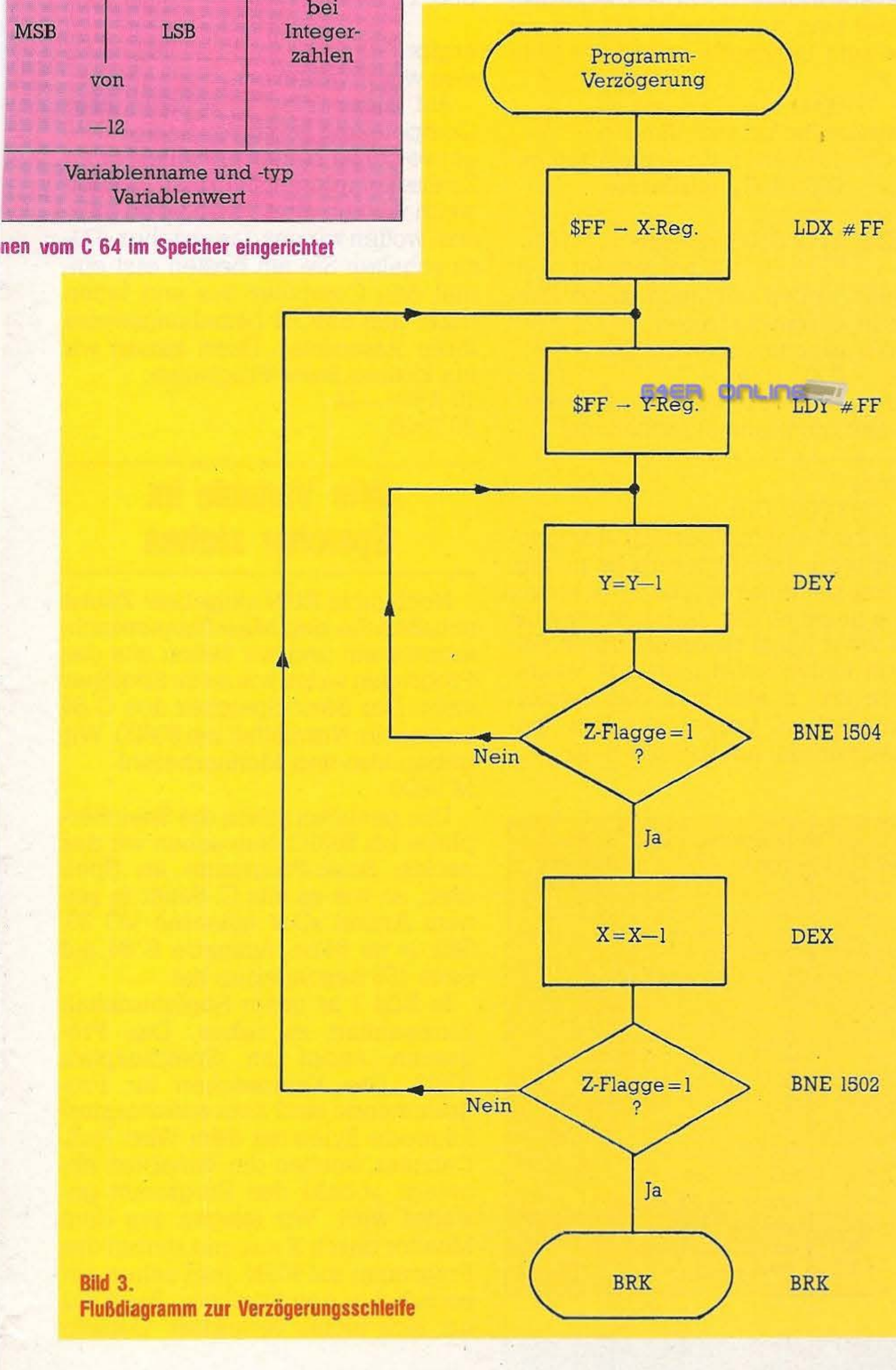


Bild 3. Flußdiagramm zur Verzögerungsschleife

Damit sollen Sie dann auch noch gleich zwei neue Befehle kennenlernen: SED und CLD. Der erstere hat nichts mit Parteien zu tun, sondern ist die Abkürzung für »Set dezimal-flag«, also setze die Dezimalflagge. So schalten Sie den Dezimal-Modus ein. Wie Sie sicher schon messerscharf geschlossen haben, heißt CLD »Clear dezimal-flag«, also setze die Dezimalflagge auf Null, wodurch dieser Modus wieder auszuschalten ist.

Wichtig! Wenn Sie argwöhnen, daß in einem Programm irgendwann mal die Dezimal-Flagge gesetzt sein könnte, dann gehen Sie auf Nummer sicher und schieben vor eine Rechenoperation, die nicht im Dezimalmodus laufen soll, ein CLD.

Beide Befehle sind 1-Byte-Befehle mit implizierter Adressierung. Sie beeinflussen lediglich die Dezimalflagge.

Das Geheimnis der negativen Binärzahlen

Wie schon mal betont: Der Computer ist strohdumm. Er kann nicht einmal auf normale Weise voneinander abziehen! Deswegen geht er den komplizierten Weg: Er addiert eine negative Zahl. Nur: Wie sehen negative Binärzahlen aus? Wir werden diese Frage in drei Etappen beantworten.

a) Man könnte eine Flagge setzen, die 1 ist bei negativen und 0 bei positiven Zahlen. Bei einigen Fließkommazahlen wird das auch so gemacht. Hier aber setzt man die Flagge direkt in die Zahl ein: Bit 7 jeder Zahl ist jetzt ein Vorzeichenmerkmal. Wenn dieses Bit 0 ist, handelt es sich um eine positive, wenn es 1 ist, um eine negative Zahl. Auf diese Weise ist also +1 wie bisher 0000 0001, wohingegen —1 jetzt 1000 0001 hieße. Damit wird allerdings der Zahlenbereich, der durch ein Byte auszudrücken ist, verschoben. 255=binär 1111 1111 kann so nicht mehr verwendet werden. Die größte Zahl, die jetzt ausgedrückt werden kann, ist 0111 1111 = dezimal 127. Die kleinste Zahl ist dann 1111 1111 = —127. Probieren wir mal aus, wie sich damit rechnen läßt:

```
+10   0000 1010
—6    1000 0110
————————————— +
ergibt 1001 0000  = —16,
```

was offensichtlich falsch ist, denn nach Adam Riese sollte +4 heraus-

Assembler ist keine Alchimie

kommen. So kann man also nicht rechnen!

Man nennt diese Art der Zahlendarstellung, übrigens »signed binary«-Format, also in Deutsch: markierte Binärzahlen.

b) Der nächste Schritt ist das sogenannte Einerkomplement. Dabei tritt für die positiven Zahlen keine Änderung ein. Die negativen entstehen aus den positiven durch Komplementbildung, das heißt jedes Bit der positiven Zahl wird in sein Gegenteil verkehrt, wie es das folgende Beispiel zeigen soll:

0000 1100 ist +12,

dann ist das Einerkomplement:

1111 0011 = —12.

Komplement ist nicht kompliziert

Interessanterweise taucht hier auch wieder das Merkmal der »signed binary«-Zahlen auf: die 1 in Bit 7 bei negativen Zahlen. Beschränkt man sich auf den Zahlenbereich, der für die »signed binary«-Zahlen gültig war, dann hätten wir jetzt beide Darstellungsweisen miteinander vereint. Nun müssen wir natürlich noch feststellen, ob man so auch rechnen kann.

```
+8    0000 1000
—6    1111 1001
  in Einerkomplementdarstellung
————————————— +
ergibt (1) 0000 0001
```

was 1 mit einem Übertrag ergäbe, jedenfalls nicht 2, wie's sich gehört. Also ist auch die Einerkomplementdarstellung noch nicht das Gelbe vom Ei.

c) Ich will Sie nicht länger auf die Folter spannen: Wenn man zum Einerkomplement einer Zahl noch 1 dazuzählt, erhält man das Zweierkomplement. Und genau so werden negative Zahlen in unserem Computer gehandhabt. Die positiven Zahlen bleiben unverändert. Von den negativen bildet man das Zweierkomplement wie zum Beispiel hier mit der Zahl —12:

```
12      0000 1100
normale Binärdarstellung
(—12)   1111 0011  Einerkomplement
+1      0000 0001  addieren
————————————————
—12     1111 0100  Zweier-
                   komplement
```

Jetzt wollen wir auch diese Zahlenart ausgiebig testen:

Wir rechnen nochmal 8-6:

```
+8    0000 1000
—6    1111 1010 das ist —6 in der
Zweierkomplementdarstellung.
————————————— +
ergibt
   (1) 0000 0010
```

also 2 mit einem Übertrag, der ignoriert wird. Das Ergebnis ist richtig. Wenn bei einer solchen Rechnung eine negative Zahl herauskommt, ist sie nicht leicht zu erkennen. In solchen Fällen kehrt man das Vorzeichen um, indem man das Zweierkomplement berechnet. Das machen wir mal am Beispiel 5—6:

```
+5    0000 0101
—6    1111 1010
```

das ist wieder unser Zweierkomplement von 6, also —6

```
————————————— +
ergibt 1111 1111
```

das ist —1 in der Zweierkomplementdarstellung. Zur Kontrolle nun die Vorzeichenumkehr durch Umrechnen ins Zweierkomplement:

```
Einerkomplement davon  0000 0000
plus 1                 0000 0001
                       ———————— +
ergibt                 0000 0001
```

also wie erwartet +1.

Auf diese Weise rechnet unser Computer mit negativen Zahlen. Negative ganze Zahlen speichert er im Zweierkomplement-Format. Auch wenn wir nun etwas vorgreifen müssen, wollen wir uns das ansehen. Dazu schalten Sie am besten erst einmal den Computer aus und laden dann den SMON beziehungsweise ihren Assembler. Dann bauen wir ein kleines Basic-Programm:

```
10 A%=—12
20 END
```

Wie Variable im Speicher stehen

Noch nicht RUN eingeben! Zuerst schalten Sie den Maschinensprachmonitor ein und wir sehen uns das Programm so an, wie es im Speicher steht. Der Basic-Speicher des C 64 beginnt im Normalfall bei $0800. Wir geben also den Monitorbefehl

M 0800

Uns genügen schon die Speicherplätze bis $081C. Nun sehen wir das nackte Basic-Programm im Speicher, so wie es uns C. Sauer in seinem Artikel »Der gläserne VC 20, Teil 1« im 64'er, Ausgabe 9/84 auf Seite 156 beschrieben hat.

In Bild 1 ist unser Speicherinhalt kommentiert zu sehen. Das Programm endet im Speicherplatz $0813. Das Kennzeichen für Programmende sind zwei aufeinanderfolgende Bytes mit dem Wert Null. Dahinter werden die Variablen abgelegt, sobald das Programm gestartet wird. Wir steigen aus dem Monitor durch X aus und starten das Programm mit RUN. Jetzt sehen wir nochmal in den Speicher. Bis $0813 hat sich nichts verändert. Danach

Befehlswort	Adressierung	Byteanzahl	Code Hex	Code Dez	Dauer in Taktzyklen	Beeinflussung von Flaggen
INX	implizcit	1	E8	232	2	N,Z
INY	implizit	1	C8	200	2	N,Z
INC	absolut	3	EE	238	6	N,Z
DEX	implizit	1	CA	202	2	N,Z
DEY	implizit	1	88	136	2	N,Z
DEC	absolut	3	CE	206	6	N,Z
SED	implizit	1	F8	248	2	1 → D
CLD	implizit	1	D8	216	2	0 → D
BNE	relativ	2	D0	208	2 +1 bei Verzweigung +2 bei Überschreiten einer Seitengrenze	—

Tabelle. Die in dieser Folge erwähnten Befehle

Teil 3

aber ist jetzt in 7 Bytes die Variable A% abgelegt. Das zeigt Bild 2.

Zunächst einmal die Bytes $0814 und $0815: Hier wird der Variablenname und der -typ angegeben. Der Typ ist aus den Bits 7 zu erkennen. Sind beide (wie hier) gleich 1, dann handelt es sich um eine Integervariable (also eine ganze Zahl). Läßt man die Kennbits außer acht, zeigt sich, daß in $0814 der Code für den Buchstaben A steht und $0815 nur den Wert 0 enthält. Nun zum Rest: Der C 64 legt Integers in nur 2 Bytes ab — die restlichen 3 Bytes $0818 bis $081A bleiben unbenutzt. Das ist auch dann der Fall, wenn danach noch weitere Variable kommen. Es bringt also keine Speicherersparnis (VC 20-Benutzer aufgepaßt!), wenn man mit Ganzzahlvariablen arbeitet!

In $0817 steht $F4, welches binär ausgedrückt 1111 0100 ist. Das kennen wir noch von weiter oben als die —12 im Zweierkomplement-Format. Woher kommt $FF in Speicherzelle $0816? Wie gesagt, die Integers werden in 2 Bytes gespeichert, und wenn wir —12 in 16 Bits ausdrücken, dann sieht das so aus:

```
+12               0000 0000 0000 1100
Einerkomplement:
                  1111 1111 1111 0011
plus 1
                  0000 0000 0000 0001
-------------------------------------
ergibt —12:       1111 1111 1111 0100
                  MSB             LSB
                  =$FF            =$F4
```

als 16-Bit-Zweierkomplement.

Die größte positive ganze Zahl, die man in 2 Bytes ausdrücken kann, ist 32767, was binär
0111 1111 1111 1111
ergibt. Die kleinste ist
1000 0000 0000 0000
also —32768. Das ist der Grund dafür, daß der C 64 Integers größer als 32767 oder kleiner als —32767 dankend mit ILLEGAL QUANTITY ERROR ablehnt, wenn sie als Argument verwendet werden. (Die Zahl —32768 kann als Ergebnis von logischen Operationen durchaus auftauchen.)

Damit will ich Sie für diesmal von den Zahlenspielereien erlösen. In der nächsten Folge müssen wir darauf nochmal zurückkommen. Sie können die Art des Abziehens von Zahlen durch Addieren des Zweierkomplementes bis zum nächsten Mal an weiteren Beispielen üben. Wenn Sie das mit 16-Bit-Zahlen tun, werden Sie bald feststellen, daß noch nicht alles so funktioniert wie es sollte...

Wir können jetzt übrigens auch das Rätsel lösen, weshalb bei positiven Zahlen (zum Beispiel LDA #FF) die Negativ-Flagge auf 1 geht: Die Flagge wird immer dann gezückt, wenn eine Zahl auftritt, die in Bit 7 eine 1 aufweist. Ganz einfach, gell?

Ein wirkungsvolles Zweiglein: BNE

Vermutlich raucht Ihnen nach soviel Zahlensalat der Kopf. Deshalb sollen Sie zur Entspannung noch einen neuen Assembler-Befehl kennenlernen und auch gleich ein nützliches Programmbeispiel dazu.

BNE heißt »branch if not equal zero«, was man übersetzen kann mit »verzweige, wenn ungleich Null«. Genauer gesagt: Es wird dann verzweigt — also zu einer angegebenen Adresse gesprungen —, wenn die Z-Flagge (die haben wir bei den INX,DEX...-Befehlen genauer untersucht) nicht gesetzt ist, also 0 zeigt. Sehen wir uns das mal an der nachfolgenden Verzögerungsschleife an, deren Flußdiagramm Bild 3 zeigt.

Das Programmchen dazu:

```
1500    LDX #FF
1502    LDY #FF
1504    DEY
1505    BNE 1504
1507    DEX
1508    BNE 1502
150A    BRK
```

Zunächst einmal werden das X- und das Y-Register als Zähler initialisiert (also mit einem Ausgangswert geladen). Mit dem vorhin behandelten Befehl DEY wird dann das Y-Register um 1 heruntergezählt, was jetzt $FE ergibt. Für die Nullflagge (Z) bedeutet das den Inhalt 0, denn es liegt kein Grund vor, sie zu setzen (also eine 1 dort anzuzeigen), weil noch keine Null aufgetreten ist. Bei der nachfolgenden Prüfung durch BNE wird also eine Verzweigung nach 1504 das Ergebnis sein, worauf das Y-Register weiter verringert und dann die Z-Flagge erneut geprüft wird und so weiter. Das geht so lange, bis nun wirklich endlich die Null im Y-Register erreicht ist. In diesem Fall zählt DEX nun das X-Register herunter und der nächste BNE-Befehl führt zum Sprung nach 1502, wo das Y-Register wieder auf $FF gesetzt wird. Auf diese Weise wird die äußere Schleife 255mal und die innere 65025mal durchlaufen.

Kein Widerspruch: Assembler-Programme langsamer machen

Sie haben beim Eingeben des Programmes vermutlich etwas gestutzt, als der Assembler nach dem BNE 1504 als nächste Adresse statt dem erwarteten 1508 eine 1507 ausgegeben hat. Der Befehl sieht zwar wie ein 3-Byte-Befehl aus, ist aber nur ein 2-Byte-Befehl! Das liegt an der speziellen Art der Adressierung von solchen Branch-Anweisungen: Der sogenannten relativen Adressierung, die wir aber erst später mit den anderen Branch-Befehlen behandeln werden.

Wenn Sie das Programm mit G 1500 starten, werden Sie — obwohl alles in Maschinensprache schnell läuft — eine merkliche Verzögerung feststellen, bevor die Registeranzeige auftaucht. Noch längere Verzögerungen lassen sich ohne weiteres erreichen, indem man mehr Schleifen ineinanderschachtelt. Dabei verwendet man dann den DEC-Befehl.

In der Tabelle sind auch die Zyklen angegeben, die die heute neu gelernten Befehle zur Abarbeitung benötigen. Mit solchen Angaben lassen sich recht genau definierte Zeiten einstellen, in denen der Computer nichts anderes tut als durch das Programm zu flitzen. Wozu das dient, braucht wohl kaum noch gesagt werden: Wenn Sie zum Beispiel einen Text auf dem Bildschirm lesen wollen, bevor das Programm weiterläuft oder wenn Sie mit Peripherie arbeiten, die langsamer als das Programm ist oder... Allerdings muß noch gesagt werden, daß es noch elegantere Methoden zur Verzögerungs-Programmierung gibt als das Lahmlegen des Computers, aber dazu kommen wir erst in einer späteren Folge. (Heimo Ponnath/gk)

Der gläserne VC 20 Teil 3

In der letzten Folge befaßten wir uns schwerpunktmäßig mit der Zeropage. Diesmal wird der sich daran anschließende Adreßbereich von $0100 bis $03FF unter die Lupe genommen.

Dieser Bereich ist so interessant, daß sich die Betrachtungen darüber bis in die 4. Folge erstrecken werden. Übrigens ist dieser Teil auch auf den C 64 anwendbar, denn Betriebssytem und Basic-Interpreter dieser beiden Computer sind ja nahezu identisch.

Wenn wir einen Basic-Befehl im Direktmodus (LIST, RUN, PRINT) oder eine Programmzeile mit Zeilennummer eingeben, werden die Informationen — wie bekannt — auf den Bildschirm geschrieben und gelangen gleichzeitig in den Basic-Eingabepuffer (Adresse 512-600/ $0200-$0258). Dort werden Programmzeilen oder direkte Befehle zunächst einmal im ASCII-Format gesammelt (Bild 1a). Dies geschieht solange, bis man die RETURN-Taste betätigt.

Dieser Puffer hat eine Kapazität von 88 Zeichen — also den bekannten vier Bildschirmzeilen.

Der Weg einer Eingabezeile

Drückt man die RETURN-Taste, so beginnt der Interpreter mit der Auswertung der Kommandos. Eingaben ohne Zeilennummer werden auf ihre Syntax hin überprüft und danach ausgeführt.

Verfolgen wir nun einmal genauer den Weg einer Befehlszeile. Die einzelnen ASCII-Zeichen werden von der inzwischen hinreichend bekannten CHRGET-Routine aus dem Puffer gelesen und mit den Befehlswörtern aus dem ROM verglichen. War die Überprüfung positiv — ist der Befehl als identifiziert worden —, so verkürzt der Computer die Kommandozeile, indem er die Befehle in Token (siehe Teil 1, Tabelle 1) umwandelt. Diese Prozedur durchlaufen sowohl die Programmzeilen (mit Zeilennummer) als auch die direkten Kommandos (Bild 1b). An dieser Stelle trennen sich nun aber die Wege dieser beiden Zeilentypen.

Zunächst zu dem weiteren Weg einer Programmzeile. Die inzwischen komplett übersetzte Zeile im Basic-Puffer wird nun in einem zweiten Durchlauf vom Zwischenspeicher in den Programmspeicher übertragen, wobei sie auch gleich richtig eingeordnet wird (damit die Reihenfolge der Zeilennummern stimmt). Weil sich dadurch der Programmbereich im Basic-Speicher vergrößert, muß der Variablenbereich weiter oben angesiedelt werden. Die bis dahin gespeicherten Variablen werden dadurch natürlich überschrieben. Nach dem Übertragen der Programmzeile springt das Interpreterprogramm wieder in die Warteschleife zurück, damit weitere Befehle entgegen genommen werden können.

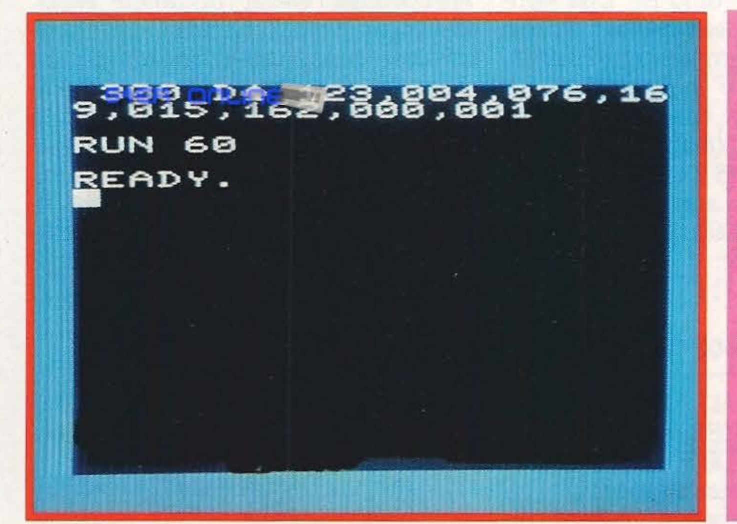


Bild 2. Anhand dieses Bildschirmfotos muß der Cursor programmiert werden

Nun möchte ich den weiteren Weg einer Direktmoduszeile beschreiben, denn der verläuft anders. Nachdem die Zeile mit Hilfe der CHRGET-Routine in Interpretercode (also Token) umgewandelt worden ist, wird der 2-Byte-Zeiger ($7A-$7B) auf den Pufferanfang zurückgestellt und der Inhalt wie eine Programmzeile behandelt und abgearbeitet (wieder mit Hilfe der CHRGET-Routine).

Verbotenes

Die Befehle INPUT und GET dürfen im Direktmodus nicht verwendet werden. Der Grund liegt darin, daß diese Eingabebefehle ebenfalls den Basic-Eingabepuffer zur Zwischenspeicherung der Daten verwenden. Das »Direktmodusprogramm«, was sich zu dieser Zeit im Puffer befindet, würde dann überschrieben. Um das zu verhindern, sind diese Kommandos im Direktmodus verboten (Fehlermeldung »ILLEGAL DIRECT«).

Die Tastaturverwaltung

Die Schlüsselfunktionen für Basic wie beispielsweise die Umwandlung der ASCII-Zeichen in Token (wie eben beschrieben), Umwandlung der Token in Klartext (die Befehlswörter werden bei LIST wieder als ASCII-Zeichen sichtbar gemacht), werden durch indirekte Sprünge über Vektoren abgewickelt. Durch Zusatzroutinen besteht so die Möglichkeit, Klartextbefehle wie SOUND, KEY, OLD ect. in den Interpreter mit einzubinden. Dieses Verfahren besprechen wir später, zunächst aber schreiten wir in der Betrachtung des Adreßbereiches von $0277-$03FF fort (dieser ist in Tabelle 1 genauer aufgelistet).

Der nächste, größere Komplex, den wir hier behandeln wollen, ist die Tastaturverwaltung. Wie sie bestimmt schon öfter bemerkt oder gelesen haben, wickelt der VC 20 (der C 64 übrigens auch) seine Tastaturoperationen ebenfalls über einen Puffer ab. Dies wird beim LISTen von Basic-Programmen deutlich, denn während der Computer ein Programm ausgibt, können sie alle Tasten (mit Ausnahme des STOP-Keys) drücken — diese erscheinen aber nicht auf dem Bildschirm. Erst wenn das Programm zu Ende gelistet ist, sieht man, daß keine Taste »vergessen« wurde, denn alle Einga-

ben wurden im Tastaturpuffer (Adresse 631-640/ $0277-$0280) zwischengespeichert.

Dieser Puffer hat eine Kapazität von maximal zehn Zeichen. Wem dies zuviel ist, der kann die Länge des Puffers durch Adresse 649 einstellen. Ratsam kann dies bei relativ langsamen Basic-Spielen sein, wo es empfehlenswert ist, nur ein Zeichen im Tastaturpuffer zwischenzuspeichern (POKE 649,1). Anderenfalls »hinkt« die Tastatur immer den Ereignissen hinterher.

Mit Hilfe des Tastaturpuffers kann aber auch — und das ist das interessante an diesen zehn Bytes — eine relativ unkonventionelle Art der Programmierung praktiziert werden.

Dezimal	Hexadezimal	Bemerkung
255-266	00FF-010A	Arbeitsspeicher für Fließkomma nach ASCII
256-318	0100-013E	Korrekturpuffer für Bandbetrieb
256-511	0100-01FF	Prozessor Stack
512-600	0200-0258	Basic-Eingabepuffer
601-610	0259-0262	Tabelle der logischen Filenummern...
611-620	0263-026C	....sowie der dazugehörigen Gerätenummern....
621-630	026D-0276	...und der entsprechenden Sekundäradressen
631-640	0277-0280	Tastaturpuffer
641-642	0281-0282	Start des verfügbaren RAM-Bereichs
643-644	0283-0284	Ende des verfügbaren RAM-Bereichs
645	0285	Timeout-Flag für den seriellen Port
646	0286	Aktueller Farbcode
647	0287	Farbe unter dem Cursor
648	0288	High-Byte des Bildschirmspeichers (Information für das Betriebssystem)
649	0289	Größe des Tastaturpuffers (Maximum 10)
650	028A	Repeat-Flag (0: Cursor + Space/ 64: Keine Taste/ 128: Alle Tasten mit Repeat)
651	028B	Repeat-Zähler (bestimmt die Wartezeit bis die Taste wiederholt wird)
652	028C	Repeat-Verzögerung (bestimmt die Zeit, bis die Taste das erste mal wiederholt wird)
653	028D	Kontrolltasten-Flag (1: SHIFT/ 2: CBM/ 4: CTRL. Es können auch 2 Tasten erkannt werden zum Beispiel 5: SHIFT + CTRL)
654	028E	Letzte Kontrolltaste (Identisch mit 653)
655-656	028F-0290	Vektor für Tastaturdecodierung
657	0291	Flag für SHIFT + CBM gesperrt (keine Groß-Kleinschrift Umschaltung. 0: Normal/ 128: Sperre)
658	0292	Flag für Scrolling
659-670	0293-029E	RS 232 Register, Zeiger, ect.
671-672	029F-02A0	Zwischenspeicher für IRQ bei Bandbetrieb
673-676	02A1-02A4	Diverse VIA-Zeiger
677-767	02A5-02FF	Zwischenspeicher einer Bildschirmzeile
786-769	0300-0301	Vektor für Fehlermeldung (C43A)
770-771	0302-0303	Vektor für Basic-Warmstart (C483)
772-773	0304-0305	Vektor für Umwandlung von ASCII in Token (C579)
774-775	0306-0307	Vektor für Umwandlung von Token in ASCII (C717)
776-777	0308-0309	Vektor für Basic-Befehlsadresse (C7E1)
778-779	030A-030B	Vektor für arithmetisches Element (CE83)
780	030C	Akku für SYS-Befehl (Bei SYS wird 780 in den Akku geladen)
781	030D	X-Reg für SYS-Befehl
782	030E	Y-Reg für SYS-Befehl
783	030F	Speicher für Status Register für SYS-Befehl
788-789	0314-0315	IRQ-Vektor (EABF) *** Kernal-Vektoren
790-791	0316-0317	BRK-Vektor (FED2)
792-793	0318-0319	NMI-Vektor (FEAD)
794-795	031A-031B	OPEN-Vektor (F40A)
796-797	031C-031D	CLOSE-Vektor (F34A)
798-799	031E-031F	Kanal für Eingabe (CHKIN, F2C7)
800-801	0320-0321	Kanal für Ausgabe (CHOUT, F309)
802-803	0322-0323	Kanäle initialisieren (CLRCH, F3F3)
804-805	0324-0325	Eingabe-Vektor (F20E)
806-807	0326-0327	Ausgabe-Vektor (F27A)
808-809	0328-0329	Vektor für STOP-Taste prüfen (F770)
810-811	032A-032B	GET-Vektor (F1F5)
812-813	032C-032D	Alle Files schließen ((F3EF)
814-815	032E-032F	User-Vektor (FED2)
816-817	0330-0331	LOAD—Vektor (F549)
818-819	0332-0333	SAVE-Vektor (F685)
828-1019	033C-03FB	Kassettenpuffer

Tabelle 1. Die Adreßbelegung der Seiten 2 bis 4 beim VC 20

Der »DATA-Erzeuger«

Um dies verständlich zu machen, habe ich ein Programm (Listing 1) geschrieben, welches DATA-Zeilen aus Maschinenprogrammen oder Sonderzeichen erzeugt. Die Problematik dabei ist folgende: Wenn ich DATA-Zeilen ins Programm schreiben möchte, so ist dies nur im Direktmodus möglich. Um dies automatisch zu tun, benötigen wir ein Programm. Also was tun? — Man bedient sich des Tastaturpuffers!

Dazu nochmals die Durchleuchtung der Funktionsweise. Während des Systeminterrupts (was dort geschieht klären wir in einer der nächsten Folgen), der alle 60stel Sekunde durchlaufen wird, fragt der Computer die Tastatur ab. Ein eingegebenes Zeichen wird dabei im Tastaturpuffer abgelegt, wo es so lange verbleibt, bis ein Zeichen von der Tastatur benötigt wird. Dies ist zum Beispiel im Direktmodus der Fall (wenn der Cursor blinkt) oder im Programm — bei INPUT oder GET. Die Zeichen werden dann wieder aus dem Tastaturpuffer hervorgeholt und zwar nach dem Prinzip »First in, First out«. Bei unserem Programm werden zunächst einmal sechs Speicherzellen initialisiert. Die ersten zwei Adressen enthalten die Anfangsadresse der abzuspeichernden Daten aus dem Speicher. Dieser Zeiger wird solange inkrementiert, bis er den Wert der Endadresse — der in den zwei folgenden Bytes abgelegt ist — erreicht hat. In den letzten zwei Speicherstellen (Adresse 252, 253) steht die Zeilennummer in der Reihenfolge Low-/High-Byte. Bequemer währe es natürlich, wenn man normale Variablen verwenden könnte. Dies ist jedoch nicht möglich, weil diese beim Einfügen einer DATA-Zeile gelöscht werden. Darum bleibt nur der Umweg über Speicherstellen, deren Inhalte durch Basic nicht überschrieben werden können.

Als nächstes erzeugt das Programm — mit Hilfe der in 252/253 gespeicherten Zwei-Byte-Zahl — eine Zeilennummer, welche auf den Bildschirm gePRINTet wird. Dann schreibt es das Befehlswort »DATA«

und druckt acht dreistellige Zahlen (die Daten aus dem zu verarbeitenden Maschinenprogramm) aus. Nun haben wir eine fertige Programmzeile auf dem Bildschirm stehen, die sich allerdings noch nicht im Speicher befindet. Dies erledigt jetzt unser Tastaturpuffer.

Vorher müssen wir und jedoch genau überlegen, wie unser Bildschirm aussieht, denn dementsprechend muß der Cursor programmiert werden.

Cursorprogrammierung einmal anders

Bild 2 zeigt ein Bildschirmfoto dieser Situation. Zuerst wird der Cursor mit HOME (= CHR$ (19)) an die linke obere Ecke befördert. Dort wird durch einen Druck auf die RETURN-Taste (= CHR$(13)) die Basic-Zeile in den Speicher übernommen. Nun befindet sich der Cursor in der dritten Zeile. Durch ein »Cursor down« (= CHR$(17)) bewegt er sich eine Zeile nach unten und steht nun auf dem Befehlswort RUN 60. Ein weiteres RETURN bewirkt den erneuten Start des Hilfsprogramms.

Wir benötigen für die Cursorbewegung also vier Werte. Diese werden vor dem Ende des Programms mit »POKE 631,19: POKE 632, 13: POKE 633,17: POKE 634, 13: POKE 198,4« in den Tastaturpuffer geschrieben. Hierdurch simulieren wir eine gedrückte Tastenfolge: denn nachdem sich der Computer über den Befehl END wieder im Direktmodus befindet, wird zuerst der Tastaturpuffer geleert, wodurch der Cursor den vorbestimmten Weg nimmt. POKE 198,4 gibt an (das noch als Nachtrag) wieviele Zeichen sich momentan im Puffer befinden. Dieser POKE-Befehl darf bei der Manipulation des Tastaturspeichers nie vergessen werden. Ferner ist die Bereitschaftsmeldung »READY«, die beim Übergang in den Direktmodus ausgegeben wird, zu berücksichtigen. Man muß beachten, daß dadurch nicht die bereits auf dem Bildschirm befindlichen Zeichen überschrieben werden.

Die Bedienung des Programms an sich ist ganz einfach. Nach Eingabe der Anfangs- und Endadresse werden die DATA-Zeilen mit jeweils acht Elementen in den Programmspeicher generiert; begonnen wird mit Zeilennummer 300, die in 10er-Schritten erhöht wird. Da sich das Listing selbst kommentiert, erübrigt sich alles Weitere. Der Vollständigkeit halber ist noch zu erwähnen, daß die REM-Zeilen nicht mit eingegeben werden müssen.

```
1 REM DATA-ERZEUGER
2 REM
10 INPUT"STARTADRESSE ";SA
20 INPUT"ENDADRESSE   ";EA
30 A=SA:GOSUB250:POKE248,A1:POKE249,A2
40 A=EA:GOSUB250:POKE250,A1:POKE251,A2
50 POKE252,44:POKE253,1
55 REM *** STARTZEILENNUMMER = 300
60 DEF FNZ(X)=PEEK(X)+PEEK(X+1)*256
65 REM *** DOPPELPEEK FUNKTION
70 AD=FNZ(252)
75 REM *** AD = AKTUELLE ZEILENNUMMER
80 PRINT""AD"D ";
90 FORT=0TO7
95 REM *** DATA ZEILE MIT 8 ELEMENTEN
96 REM ***          ERZEUGEN
100 B=PEEK(FNZ(248)+T)
110 B$=STR$(B):L=LEN(B$)
120 B$=RIGHT$(B$,L-1)
130 IFL=4THEN150
140 FORY=L+1TO4:B$="0"+B$:NEXT
145 REM *** JEDES ELEMENT IST 3 STELLIG
150 B$=","+B$
160 IFT=0THENB$=RIGHT$(B$,3)
170 PRINTB$;:IFFNZ(248)+T>=FNZ(250)
    THEN230
175 REM *** LFD. ADRESSE > END ADRESSE ?
180 NEXT
190 A=AD+10:GOSUB250:POKE252,A1:
    POKE253,A2
200 A=FNZ(248)+8:GOSUB250:POKE248,A1:
    POKE249,A2
210 PRINT:PRINT"RUN 60"
213 REM *** TASTATURPUFFER MIT <HOME>,
215 REM *** <RETURN>, <CRSR DOWN> UND
217 REM ***     <REUTRN> FUELLEN
220 POKE631,19:POKE632,13:POKE633,17:
    POKE634,13:POKE198,4
230 END
250 A2=INT(A/256):A1=A-A2*256
260 RETURN

READY.
```

Listing 1. Der DATA-Erzeuger

Benutzt wurde der Tastaturpuffer bereits in einem Programm, das im ersten Teil dieser Serie abgedruckt wurde. Die Rede ist von der Autostartroutine, welche einen Basic-Programmstart (aus einem Maschinenprogramm heraus) durch Füllen des Puffers mit dem Kommando RUN (+ CHR$ (13)) realisierte.

Die Eckadressen

Als nächstes besprechen wir die Adressen 641-644/ $0281-$0284. Sie enthalten die Anfangs- beziehungsweise Endadresse des verfügbaren Speicherbereiches.

Bei dem Systemreset ($FD22/ 64802) werden verschiedene Routinen wie beispielsweise Initialisierung der Zeropage, Setzen der Vektoren, RAM-Test ect. durchlaufen. Dabei wird unter anderem auch der verfügbare Speicherplatz festgestellt und die Eckadressen den obengenannten Registern übergeben. Diese Daten sind die Grundlage für alle weiteren Grundeinstellungen des Systems, also Basic-Beginn und -Ende, Beginn des Video- und Farbspeichers ect.

Wird es nun nötig, eine Speicherkonfiguration zu simulieren, beispielsweise Grundversion bei eingesteckter 8-KByte-Erweiterung, so kann das über diese Zeiger geschehen:

POKE 642, 16: POKE 644, 30: SYS 64970: SYS 64821 bewirkt diese Simulation. Das Unterprogramm im Betriebssystem (Adresse 64970), das zur Reset-Routine gehört, hat die Aufgabe, die Seiten 0 (Zeropage), 2 und 3 zu löschen. Danach prüft der Computer seine Speicherzellen. Dieser Test hat die Aufgabe, das RAM auf seine Funktionsfähigkeit

```
1 REM FUNKTIONSTASTEN
2 REM
3 REM  (BASIC-LADER)
4 REM
100 DIMS(2)
110 S(0)=18372:S(1)=16820:S(2)=7987
120 PRINT"FUNKTIONSTASTEN:"
130 FORY=0TO2:PS=0
140 FORT=0TO151:READD:PS=PS+D:NEXT
150 IFPS<>S(Y)THENPRINT"FEHLER IN ZEIL
E":GOTO170
160 GOTO180
170 PRINT""Y*190+310"-"(Y+1)*190+310:E
ND
180 NEXT
190 POKE55,56:POKE56,PEEK(56)-2:CLR
200 AD=PEEK(56):PA=AD*256+57:RESTORE
210 FORT=PATOPA+454
220 READD
230 IFD=-1THEND=AD
240 IFD=-2THEND=AD+1
250 POKET,D:NEXT
260 PRINT"START MIT SYS"PA
270 PRINT"AKTIVIERUNG MIT SPACE"
280 GETA$:IFA$<>" "THEN280
290 SYSPA
300 SYS0L
310 DATA169,239,141,143,002,169,-01,141
320 DATA144,002,169,205,141,024,003,169
330 DATA-02,141,025,003,169,076,133,000
340 DATA169,090,133,001,169,-01,133,002
350 DATA096,032,121,000,201,076,208,052
360 DATA169,049,133,250,162,000,169,016
370 DATA133,251,169,013,032,210,255,169
380 DATA070,032,210,255,165,250,032,210
390 DATA255,169,032,032,210,255,189,077
400 DATA-02,032,210,255,232,198,251,208
410 DATA245,230,250,165,250,201,057,208
420 DATA213,076,115,000,032,121,000,201
430 DATA079,208,003,076,210,254,201,082
440 DATA208,006,032,057,-01,076,115,000
450 DATA201,044,240,003,076,008,207,032
460 DATA155,215,224,009,144,003,076,072
470 DATA210,202,134,250,032,121,000,201
480 DATA044,208,233,032,115,000,201,034
490 DATA208,226,138,010,010,010,010,170
500 DATA160,016,032,115,000,201,034,240
510 DATA009,157,077,-02,232,136,208,242
520 DATA240,009,169,000,157,077,-02,232
530 DATA136,208,249,076,115,000,165,157
540 DATA240,012,165,203,162,005,221,001
550 DATA-02,240,010,202,208,248,076,220
560 DATA235,039,047,055,063,197,197,240
570 DATA245,133,197,138,024,010,133,250
580 DATA173,141,002,201,001,240,002,198
590 DATA250,165,250,170,202,138,010,010
600 DATA010,010,133,251,168,185,077,-02
610 DATA240,016,201,094,240,015,201,039
620 DATA208,002,169,034,032,210,255,200
630 DATA208,235,169,000,133,207,240,009
640 DATA169,013,141,119,002,169,001,133
650 DATA198,076,214,235,076,073,083,084
660 DATA094,000,000,000,000,000,000,000
670 DATA000,000,000,000,082,085,078,094
680 DATA000,000,000,000,000,000,000,000
690 DATA000,000,000,000,073,078,080,085
700 DATA084,000,000,000,000,000,000,000
710 DATA000,000,000,000,071,079,083,085
720 DATA066,000,000,000,000,000,000,000
730 DATA000,000,000,000,083,089,083,048
740 DATA000,000,000,000,000,000,000,000
750 DATA000,000,000,000,082,069,084,085
760 DATA082,078,000,000,000,000,000,000
770 DATA000,000,000,000,082,069,083,084
780 DATA079,082,069,000,000,000,000,000
790 DATA000,000,000,000,076,079,065,068
800 DATA039,000,000,000,000,000,000,000
810 DATA000,000,000,000,120,072,138,072
820 DATA152,072,173,029,145,016,037,045
830 DATA030,145,170,041,002,240,026,044
840 DATA017,145,032,052,247,032,225,255
850 DATA208,018,032,082,253,032,249,253
860 DATA032,024,229,032,057,-01,108,002
870 DATA192,076,222,254,076,086,255,000

READY.
```

Listing 2. Funktionstastenbelegung (Basic-Lader)

```
>> FUNKTIONSTASTEN <<

****************** INITIALISIERUNG
1C39  LDA #$EF
1C3B  STA $028F
1C3E  LDA #$1C    ; AENDERUNG DES
1C40  STA $0290   ; TASTATUR-VEKTORS
1C43  LDA #$CD
1C45  STA $0318
1C48  LDA #$1D
1C4A  STA $0319   ; NEUER NMI VEKTOR
1C4D  LDA #$4C
1C4F  STA $00
1C51  LDA #$5A
1C53  STA $01
1C55  LDA #$1C    ; SPRUNGZEIGER
1C57  STA $02     ; FUER 'SYS0'
1C59  RTS
****************** BEFEHLSAUSWERTUNG
1C5A  JSR $0079   ; CHRGOT, LFD. ZEICHEN
1C5D  CMP #$4C    ; 'SYS0 L' ?
1C5F  BNE $1C95   ; NEIN, DANN WEITER
1C61  LDA #$31    ; SONST F1-F8 AUSLISTEN
1C63  STA $FA     ; ZAEHLER VORBEREITEN
1C65  LDX #$00
1C67  LDA #$10
1C69  STA $FB
1C6B  LDA #$0D    ; ZEILENVORSCHUB
1C6D  JSR $FFD2   ; AUSGEBEN
1C70  LDA #$46    ; 'F'
1C72  JSR $FFD2   ; AUSGEBEN
1C75  LDA $FA     ; LFD. NUMMER
1C77  JSR $FFD2   ; AUSGEBEN
1C7A  LDA #$20    ; ' '
1C7C  JSR $FFD2   ; AUSGEBEN
1C7F  LDA $1D4D,X; BEFEHLSSTRING
1C82  JSR $FFD2   ; AUSGEBEN
1C85  INX
1C86  DEC $FB
1C88  BNE $1C7F
1C8A  INC $FA
1C8C  LDA $FA
1C8E  CMP #$39    ; LETZTER STRING ?
1C90  BNE $1C67   ; NEIN, DANN WEITER
1C92  JMP $0073   ; SONST INS BASIC
1C95  JSR $0079   ; LFD. BEFEHL HOLEN
1C98  CMP #$4F    ; 'O' FUER 'SYS0 O' ?
1C9A  BNE $1C9F   ; NEIN, DANN WEITER
1C9C  JMP $FED2   ; SONST WARMSTART
1C9F  CMP #$52    ; 'R' FUER 'SYS0 R' ?
1CA1  BNE $1CA9   ; NEIN, DANN WEITER
1CA3  JSR $1C39   ; SONST PROGRAMMRESTART
1CA6  JMP $0073   ; UND ZURUECK ZU BASIC
1CA9  CMP #$2C    ; ',' FUER AENDERUNG ?
1CAB  BEQ $1CB0   ; JA, DANN WEITER
1CAD  JMP $CF08   ; SONST 'SYNTAX ERROR'
1CB0  JSR $D79B   ; TASTENNR. INS X-REG.
1CB3  CPX #$09    ; >8 ?
1CB5  BCC $1CBA   ; NEIN, DANN WEITER
1CB7  JMP $D248   ; SONST FEHLERMELDUNG
1CBA  DEX
1CBB  STX $FA
1CBD  JSR $0079   ; CHRGOT, LFD. BEFEHL
1CC0  CMP #$2C    ; ',' KOMMA ?
1CC2  BNE $1CAD   ; NEIN, DANN FEHLER
1CC4  JSR $0073   ; NAECHSTES ZEICHEN ?
1CC7  CMP #$22    ; '"' HOCHKOMMA ?
1CC9  BNE $1CAD   ; NEIN, DANN FEHLER
1CCB  TXA
1CCC  ASL
1CCD  ASL
1CCE  ASL
1CCF  ASL
1CD0  TAX         ; X-REG. * 16
1CD1  LDY #$10    ; BEFEHLSSTRING HOLEN
1CD3  JSR $0073   ; NAECHSTES ZEICHEN
1CD6  CMP #$22    ; '"' ENDE DES STRINGS
1CD8  BEQ $1CE3   ; REST MIT 0 FUELLEN
1CDA  STA $1D4D,X; SONST STRING AB-
1CDD  INX         ; SPEICHERN
1CDE  DEY
1CDF  BNE $1CD3   ; WEITER
1CE1  BEQ $1CEC   ; ENDE: JMP $0073
1CE3  LDA #$00    ; REST MIT 0 FUELLEN
1CE5  STA $1D4D,
1CE8  INX
1CE9  DEY
1CEA  BNE $1CE5
1CEC  JMP $0073   ; ZURUECK INS BASIC
****************** FUNKTIONSTASTEN ABF.
1CEF  LDA $9D     ; RUN ODER DIREKTMODUS
1CF1  BEQ $1CFF   ; BEI RUN KEINE ABFR.
1CF3  LDA $CB     ; GEDRUECKTE TASTE
1CF5  LDX #$05
1CF7  CMP $1D01,X; MIT TASTATURCODE DER
1CFA  BEQ $1D06   ; FUNKTIONSTAS. VERGL.
1CFC  DEX
1CFD  BNE $1CF7   ; WEITER,TASTE ERKANNT
1CFF  JMP $EBDC   ; TEST NEGATIV
1D02  .BYTE $27   ; < F1 >
1D03  .BYTE $2F   ; < F3 >
1D04  .BTYE $37   ; < F5 >
1D05  .BYTE $3F   ; < F7 >
1D06  CMP $C5     ; ENTPRELLUNG
1D08  BEQ $1CFF
1D0A  STA $C5
1D0C  TXA
1D0D  CLC
1D0E  ASL
1D0F  STA $FA
1D11  LDA $028D   ; KONTROLLTASTE
1D14  CMP #$01    ; <SHIFT> GEDRUECKT ?
1D16  BEQ $1D1A   ; JA, DANN WEITER
1D18  DEC $FA     ; WERT UM 1 ERNIEDR.
1D1A  LDA $FA
1D1C  TAX
1D1D  DEX
1D1E  TXA
1D1F  ASL
1D20  ASL
1D21  ASL
1D22  ASL
1D23  STA $FB     ; ACCU * 16
1D25  TAY
1D26  LDA $1D4D,Y; BEFEHLSSTRING LADEN
1D29  BEQ $1D3B   ; ENDE ?
1D2B  CMP #$5E    ; '↑' DIREKT AUSFUEHREN
1D2D  BEQ $1D3E   ; JA, DANN VERZWEIGEN
1D2F  CMP #$27    ; >'< HOCHKOMMAERSATZ?
1D31  BNE $1D35   ; NEIN, DANN WEITER
1D33  LDA #$22    ; HOCHKOMMA LADEN
1D35  JSR $FFD2   ; UND AUSGEBEN
1D38  INY
1D39  BNE $1D26
1D3B  LDA #$00    ; CURSOR ANSCHALTEN
1D3D  STA $CF
1D3F  BEQ $1D4A   ; UNBEDINGTER SPRUNG
1D41  LDA #$0D    ; <RETURN>
1D43  STA $0277   ; IN DEN PUFFER
1D46  LDA #$01    ; EIN ZEICHEN IM
1D48  STA $C6     ; TASTATURPUFFER
1D4A  JMP $EBD6   ; ZURUECK ZUM URSPRUNG
****************** BEFEHLSSPEICHER
1D4D  [illegible]
1D5D  [illegible]
1D6D  [illegible]
1D7D  [illegible]
1D8D  [illegible]
1D9D  [illegible]
1DAD  [illegible]
1DBD  [illegible]
****************** NEUE NMI ROUTINE
1DCD  SEI         ; KOPIE DER ALTEN
1DCE  PHA         ; ROUTINE
1DCF  TXA
1DD0  PHA
1DD1  TYA
1DD2  PHA
1DD3  LDA $911D
1DD6  BPL $1DFD
1DD8  AND $911E
1DDB  TAX
1DDC  AND #$02    ; RS 232 AKTIV ?
1DDE  BEQ $1DFA   ; JA, DANN VERZWEIGEN
1DE0  BIT $9111
1DE3  JSR $F734   ; STOPTASTE ABFRAGEN
1DE6  JSR $FFE1   ; STOPTASTE GEDRUECKT ?
1DE9  BNE $1DFD   ; NEIN, DANN RTI
1DEA  JSR $FD52   ; VEKTOREN SETZEN
1DEE  JSR $FDF9   ; I/O REGISTER SETZEN
1DF1  JSR $E518   ; CLR SCREEN
1DF4  JSR $1C39   ; REGISTER AENDERN
1DF7  JMP ($C002); BASIC WARMSTART
1DFA  JMP $FEDE   ; NMI FUER RS 232
1DFD  JMP $FF56   ; RTI
```

Listing 3. Funktionstastenbelegung (Assembler-Darstellung)

hin zu überprüfen, damit es nicht zu Fehlfunktionen durch einen beschädigten Speicher kommt. Bei dieser Gelegenheit wird die Ausbaustufe des Speichers festgestellt; das Ergebnis findet sich dann in den Registern für die Anfangs- beziehungsweise Endadresse. Hier steigen wir nun mit den entsprechend manipulierten Registern (642 auf 16 und 644 auf 30) in die ROM-Routine ein. Mit Hilfe dieser Werte richtet das Unterprogramm nun den Video- und Farbspeicher ein. Mit dem zweiten SYS-Befehl (SYS 64821) wird die Initialisierung fortgesetzt. Dabei richtet er den Speicher für Basic ein und gibt die Kaltstartmeldung

**** CBM BASIC V2 ****
3583 BYTES FREE

aus.

Soweit ein kleiner Einblick ins Betriebssystem, eine ausführlichere Erläuterung erfolgt in einer der nächsten Folgen.

Die Speicherorganisation

Jetzt möchte ich noch einen kleinen Einblick in die Speicherorganisation geben, was auch im Hinblick auf Grafik von Bedeutung ist. Wie hinreichend bekannt sein dürfte, gibt es drei verschiedene Speichererweiterungen zu kaufen, nämlich 3 KByte, 8 KByte und 16 KByte. Die Sammelerweiterungen (also 27/32/64 KByte-Erweiterungen sollen hier gedanklich ebenfalls in diese drei verschiedenen Module zerlegt werden.

Bei Erweiterungen von mehr als 8 KByte verändert sich die Lage des Bildschirm- und Farbspeichers (die Einstellung nimmt die Reset-Routine vor). Anhand von Bild 3 soll erläutert werden, warum eine Verschiebung notwendig ist.

Der **V**ideo-**I**nterface-**C**hip VIC (daher auch der Englische Name des VC 20), der vor allem für die Erzeugung des Fernsehsignals und den Aufbau des Bildschirmes verantwortlich ist, kann hardwaremäßig nur Videospeicherplätze zwischen 4096 und 8192 adressieren. Folglich muß das Bildschirm-RAM in diesem Bereich angesiedelt werden.

In der Grundversion liegt es zwischen Adresse 7680 und 8191 — also am Ende des verfügbaren Speichers, damit der Speicherbereich für Basic auch bei eingesteckter 3 KByte-Erweiterung durchgängig ist (läge der Bildschirmspeicher sowie bei einer 8-KByte-Erweiterung, wäre dies nicht der Fall).

Ist ein Speichermodul von mehr als 8 KByte eingesteckt (egal ob der 3-KByte-Bereich zugeschaltet ist oder nicht), so legt das System den Videospeicher an die unterste adressierbare Stelle für den VIC, also Adresse 4096. Aus diesem Grund kann die eingesteckte 3-KByte-Erweiterung nicht mehr für Basic benutzt werden, denn sonst wäre der Speicher nicht mehr durchgängig.

Programme sollten immer auf allen Erweiterungsversionen lauffähig sein. Wer also in Routinen mit dem Bildschirm- oder Farbspeicher arbeitet, kann sich mit Hilfe der Register 36866 und 36869 im VIC die

momentanen Adressen beschaffen. Bildschirm: 4*(PEEK(36866)AND128) + 64*(PEEK(36869)AND120)
Farbspeicher:
4*(PEEK(36866)AND128)+37888

Gewußt wo — Die Bildschirmadressen

Mit Hilfe bestimmter Bits aus diesen Registern bildet der VIC die Adressen, die er benötigt, um — unabhängig vom Prozessor — Bild- und Farbspeicherstellen auszulesen, damit er mit diesen Informationen das Fernsehbild erzeugen kann.

Das Betriebssystem hingegen bezieht seine Informationen über die Lage des Videospeichers nicht aus diesen VIC-internen Registern, sondern aus Adresse 648. Gibt man beispielsweise »POKE 648,28« ein, so liegt der Bildschirmspeicher zwischen 7168 und 7679. In Wirklickkeit stellt der Video-Interface-Chip — der, wie gesagt, unabhängig arbeitet — weiterhin den Speicherausschnitt zwischen Adresse 7680 und 8191 auf dem Bildschirm dar. Der Cursor schreibt also in einem ganz anderen Speicherabschnitt. Erst durch einen Warmstart (durch die Tastenkombination RUN/ STOP — RESTORE) werden die VIC-Register angepaßt.

Vom Bildschirm nun wieder zur Tastatur. In unseren systematischen Betrachtungen der Seite 1 bis 3 im Speicher, kommen wir nun zu den Adressen $0280-$0291/649-656, die der Tastatur zugeordnet sind. Sie enthalten lediglich Parameter für die Arbeit mit der Tastatur wie zum Beispiel Repeat Flag, das Flag für Kontrolltasten ect. Näheres entnehmen sie bitte Tabelle 1.

Praktisches: Die Befehlseingabe über Funktionstasten

Zwei weithin unbekannte Adressen ($028F, $0290/ 655,656) sind vorzüglich dazu geeignet, eine Funktionstastenabfrage auf Interruptbasis zu realisieren. Besagte Adressen bilden einen Vektor für die Tastaturdecodierung, der meines Erachtens nur für den Zweck der Funktionstastenabfrage geschaffen wurde.

Was sind überhaupt Vektoren? ROM, so sagt ja bereits der Name (**R**ead **O**nly **M**emory), ist grundsätzlich nicht überschreibbar. Wer dennoch das System ergänzen will (beispielsweise durch neue Basic-Befehle), ist gezwungen, das gesamte ROM auszutauschen, es sei denn, die Schöpfer des Computers haben mögliche Optionen — so wie beim VC 20 (oder C 64) — bereits eingeplant. Dies geschieht, indem aus dem ROM heraus ins RAM verzweigt wird.

Normalerweise steht an der entsprechenden RAM-Adresse nur ein Zeiger auf eine ROM-Adresse, eben ein Vektor. Durch Änderung eines solchen Vektors kann man elegant bestehende Routinen umgehen und sie durch eigene ersetzen beziehungsweise ergänzen. Soweit, so gut.

Wie im Handbuch zu lesen, gibt es Unterschiede zwischen den Bildschirmcodes eines Zeichens (»A« beispielsweise hat den Wert 1) und dem allgemein verbreiteten ASCII-Code (»A« hat hier den Wert 65). Gleiches gilt für die Tastatur. Hier unterscheidet man ebenfalls zwischen dem ASCII- und dem sogenannten Tastatur-Matrixcode. Diese VC 20-interne Codierung wird durch eine Betriebssystemroutine — die über einen Vektor verfügt (den oben erwähnten für die Tastaturdecodierung) — in ASCII-Zeichen umgewandelt. Das Maschinenprogramm in Listing 2 und 3 wird durch »verbiegen« des Vektors 655/656 in die Tastaturroutine mit eingebaut. Es fragt die Codes der Funktionstasten ab und druckt die Zeichen aus, mit denen sie belegt wurden.

Das recht komfortable Programm liegt als Basic-Lader in Listing 2 vor. Nach dem Starten mit RUN wird das

```
1 REM BEFEHLSKOPF
2 REM
100 DIMS(5)
110 REM *** PRUEFSUMMEN
120 S(0)=10538:S(1)=0:S(2)=10372
130 S(3)=14363:S(4)=12024:S(5)=733
140 FORY=0TO5:S=0
150 FORT=0TO103:READD:S=S+D:NEXT
160 IF S<>S(Y)THENPRINT"FEHLER IN T
EIL"Y+1:END
170 NEXT
180 RESTORE
190 PRINT"MODULBEREICH ODER"
200 PRINT"ENDE DES SPEICHERS ?"
210 GET A$:IFA$=""THEN210
220 IFA$="M"THENPRINT"M":GOTO380
230 IFA$<>"E"THEN210
240 PRINT"E"
245 REM *** PROGRAMM ANS SPEICHERENDE
250 POKE55,0:POKE56,PEEK(56)-3:CLR
260 AD=PEEK(56)
270 HD=AD*256:N=623
275 REM *** LESESCHLEIFE
280 FORT=HDTOHD+N
290 READD
300 IFD=-1THEND=AD
310 IFD=-2THEND=AD+1
320 IFD=-3THEND=AD+2
330 POKET,D:NEXT
340 PRINT"START DES PROGRAMMS"
350 IFFL=0THENPRINT"MIT SYS"HD"."
360 IFFL=1THENPRINT"MIT RESET (=SYS 648
02)"
370 END
375 REM *** PROGRAMM IN DEN MODULBEREICH
376 REM *** MIT IDENTIFIKATION A0CBM
380 V$="009160199254065048195194205"
390 FORE=0TO8
400 X=VAL(MID$(V$,E*3+1,3)):READD
410 POKE40960+E,X:NEXT
420 HD=40969:N=614:AD=160:FL=1:GOTO 280
430 REM TEIL 1
440 DATA120,234,234,234,234,234,234,234
450 DATA234,032,141,253,032,082,253,032
460 DATA249,253,032,024,229,162,011,189
470 DATA059,-01,157,000,003,202,016,247
480 DATA088,234,234,234,032,164,227,165
490 DATA043,164,044,032,008,196,169,071
500 DATA160,-01,032,030,203,032,018,228
510 DATA076,129,227,058,196,131,196,229
520 DATA-01,152,-02,2[illegible]02,134,206,042
530 DATA042,042,042,032,054,052,039,069
540 DATA082,032,032,066,065,083,073,067
550 DATA032,042,042,042,042,013,000,000
560 DATA000,000,000,000,000,000,000,000
570 REM TEIL 2
580 DATA000,000,000,000,000,000,000,000
590 DATA000,000,000,000,000,000,000,000
600 DATA000,000,000,000,000,000,000,000
610 DATA000,000,000,000,000,000,000,000
620 DATA000,000,000,000,000,000,000,000
630 DATA000,000,000,000,000,000,000,000
640 DATA000,000,000,000,000,000,000,000
650 DATA000,000,000,000,000,000,000,000
660 DATA000,000,000,000,000,000,000,000
670 DATA000,000,000,000,000,000,000,000
680 DATA000,000,000,000,000,000,000,000
690 DATA000,000,000,000,000,000,000,000
700 DATA000,000,000,000,000,000,000,000
710 REM TEIL 3
720 DATA000,000,000,000,000,000,000,000
730 DATA000,000,000,000,000,000,000,000
740 DATA000,000,000,000,000,166,122,160
750 DATA004,132,015,189,000,002,016,007
760 DATA201,255,240,062,232,208,244,201
770 DATA032,240,055,133,008,201,034,240
780 DATA086,036,015,112,045,201,063,208
790 DATA004,169,153,208,037,201,048,144
800 DATA004,201,060,144,029,132,113,160
810 DATA000,132,011,136,134,122,202,200
820 DATA232,189,000,002,056,249,158,192
830 DATA240,245,201,128,208,048,005,011
840 DATA164,113,232,200,153,251,001,185
850 REM TEIL 4
860 DATA251,001,240,089,056,233,058,240
870 DATA004,201,073,208,002,133,015,056
880 DATA233,085,208,159,133,008,189,000
890 DATA002,240,223,197,008,240,219,200
900 DATA153,251,001,232,208,240,166,122
910 DATA230,011,200,185,157,192,016,250
920 DATA185,158,192,208,180,160,255,202
930 DATA200,232,189,000,002,056,249,105
940 DATA-03,240,245,201,128,208,002,240
950 DATA173,166,122,230,011,200,185,104
960 DATA-03,016,250,185,105,-03,208,226
970 DATA189,000,002,016,155,076,009,198
980 DATA016,066,201,255,240,062,036,015
990 REM TEIL 5
1000 DATA048,058,170,132,073,201,204,176
1010 DATA010,160,192,132,035,160,158,132
1020 DATA034,208,011,233,076,170,160,-03
1030 DATA132,035,160,105,132,034,160,000
1040 DATA010,240,016,202,016,012,230,034
1050 DATA208,002,230,035,177,034,016,246
1060 DATA048,241,200,177,034,048,008,032
1070 DATA071,203,208,246,076,243,198,076
1080 DATA239,198,032,115,000,201,204,144
1090 DATA025,201,252,176,021,032,243,-02
1100 DATA076,174,199,233,203,010,168,185
1110 DATA009,-03,072,185,008,-03,072,076
1120 DATA115,000,032,121,000,076,231,199
1130 REM TEIL 6
1140 DATA052,253,000,000,000,000,000,000
1150 DATA000,000,000,000,000,000,000,000
1160 DATA000,000,000,000,000,000,000,000
1170 DATA000,000,000,000,000,000,000,000
1180 DATA000,000,000,000,000,000,000,000
1190 DATA000,000,000,000,000,000,000,000
1200 DATA000,000,000,000,000,000,000,000
1210 DATA000,000,000,000,000,000,000,000
1220 DATA000,000,000,000,000,000,000,000
1230 DATA000,000,000,000,000,000,000,000
1240 DATA000,000,000,000,000,000,000,000
1250 DATA000,000,000,000,000,000,000,000
1260 DATA000,075,073,076,204,000,000,000

READY.
```

Listing 4. Befehlskopf zur Definition eigener Basic-Befehle (Basic-Lader)

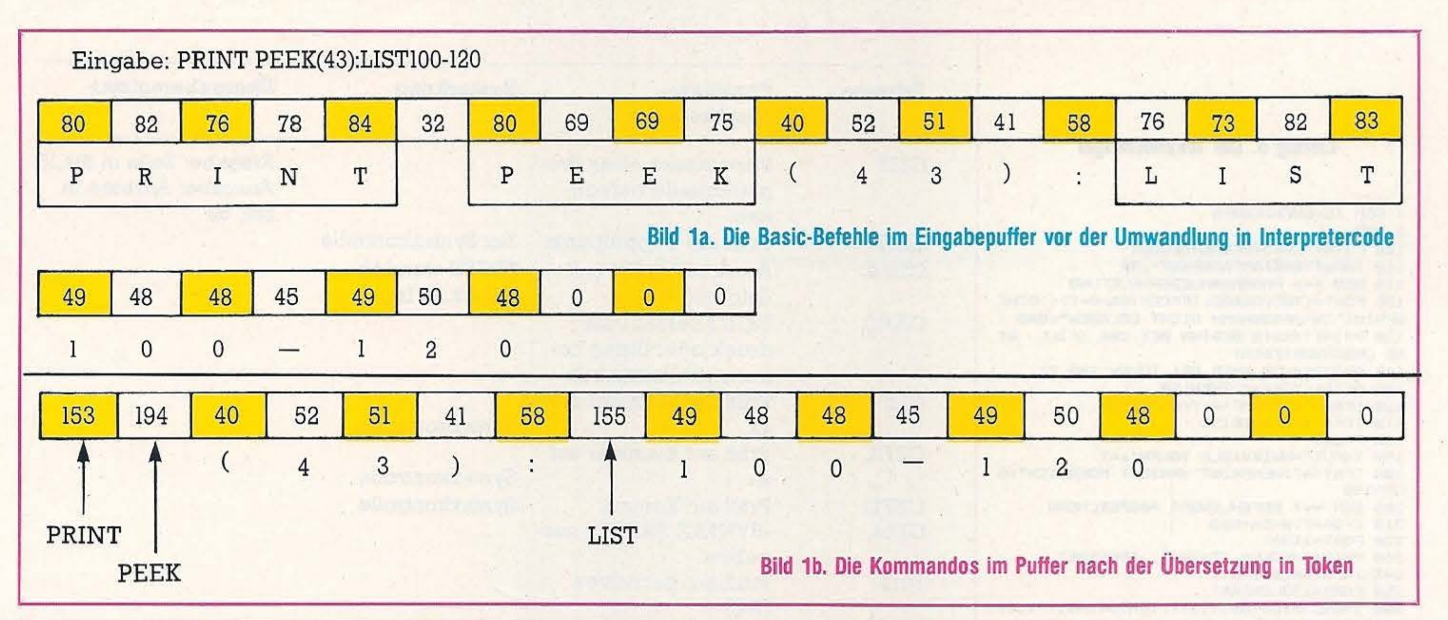


Bild 1a. Die Basic-Befehle im Eingabepuffer vor der Umwandlung in Interpretercode

Bild 1b. Die Kommandos im Puffer nach der Übersetzung in Token

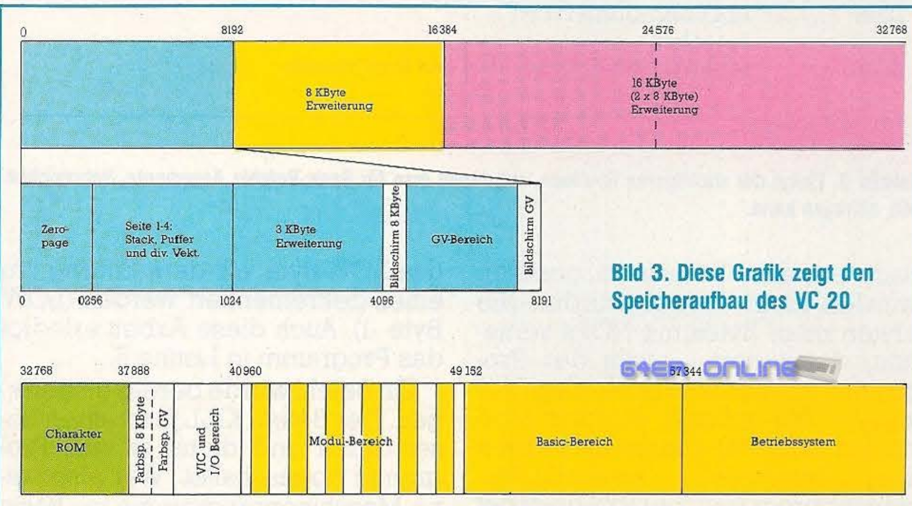


Bild 3. Diese Grafik zeigt den Speicheraufbau des VC 20



F1: LIST	— LIST-Befehl mit CHR$(13)
F2: RUN	— wie bei LIST
F3: INPUT	
F4: GOSUB	
F5: SYS0	
F6: RETURN	
F7: RESTORE	
F8: LOAD'	— ' entspricht dem Hochkomma (")

Tabelle 2. Die Grundbelegung der Funktionstasten nach dem Laden des Maschinenprogramms

Maschinenprogramm automatisch ans Ende des verfügbaren Speichers geladen. Mit der SPACE-Taste aktiviert man diese Routine. Als erstes werden die Befehle gelistet, mit denen die Funktionstasten belegt sind. (Tabelle 2).

Auffällig ist bei den Kommandos LIST und RUN der Pfeil nach oben (↑). Er bewirkt ein sofortiges Ausführen des Befehls — entspricht also »LIST« und RETURN-Taste beziehungsweise »RUN« und RETURN-Taste. Das andere auffällige Zeichen ist der Apostroph (') bei dem Kommando LOAD, der dem Hochkomma (") entspricht. Das echte Gänsefüßchen ist bereits für die Syntax des Änderungsbefehls (siehe unten) vergeben.

Auf den Befehl π wie er beispielsweise beim Programm »Basic-Switch« (Folge 2) verwendet wurde, habe ich dieses Mal aus Platzgründen verzichtet, damit Lader und Maschinenprogramm in der Grundversion Platz finden. Die Kommandos werden statt dessen über den Befehl »SYS0« (oder F5) eingegeben. Dabei machen wir uns den USR-Vektor (Adresse 0-2) als Sprungzeiger zu Nutze. Damit zur Syntax bei der Funktionstastenprogrammierung:

SYS0 L — Listen der Funktionstastenbelegungen
SYS0 O—(Off) schaltet die Funktionstasten ab.
SYS0 R — (Restart) schaltet die Funktionstasten wieder ein.
SYS0,X, »befehl« — belegt die X-te Funktionstaste mit einem Befehl.

Zum Schluß kommen wir noch zu einem Thema, mit dem ich mich mehr an den fortgeschrittenen Maschinensprachenprogrammierer wenden möchte. Im ersten Teil dieser Serie wurde beschrieben, wie der Basic-Befehlssatz mit Hilfe der CHRGET-Routine und des Befehles »π« im beschränktem Umfang erweitert werden kann. Diese Methode hat allerdings viele Unzulässigkeiten; es sind beispielsweise keine verkürzten Befehle wie beim normalen Basic (LIST = L SHIFT I) möglich.

Nun soll beschrieben werden, wie der Basic-Befehlssatz um richtige Klartextkommandos erweitert werden kann. Auch hierfür müssen wir bestehende Interpreterroutinen, die über Vektoren angesprungen werden, umgehen beziehungsweise ergänzen.

Für das Verarbeiten von Basic-Programmen sind drei Schlüsselroutinen zu substituieren. Da ist zunächst das Unterprogramm »ASCII in Token wandeln«. ($C57C) welches — wie der Name bereits sagt — die Aufgabe hat, Eingabezeilen im Basic-Puffer (wie zu Anfang beschrieben) in Interpretercode zu wandeln. Das Gegenstück dazu ist die Unterroutine »Interpretercode in Klartext wandeln.« Will man Basic-Zeilen sichtbar machen, so benutzt man das Kommando LIST. Dazu wird eben diese ROM-Routine benötigt, die die Rückumwandlung der Token in ASCII-Zeichen vornimmt.

Sämtliche Befehle sind in Form von ASCII-Zeichen im Basic-ROM enthalten, lediglich zum letzten Buchstaben jedes Befehlswortes wurde 128 ($80) addiert. Läßt man sich die Befehle durch

Listing 5. Der Tokenerzeuger

```
1 REM TOKENERZEUGER
2 REM
100 PRINT"TOKENERZEUGER"
110 INPUT"STARTADRESSE";SA
115 REM *** PROGRAMMUEBERPRUEFUNG
120 FORT=1TO3:READD:IFPEEK(SA+8+T)<>DTHE
NPRINT"PROGRAMM NICHT GELADEN":END
130 PRINT"1 BEGINN BEI 204 (KILL  WI
RD UEBERSCRIEBEN)
140 PRINT"2 ODER BEI TOKEN 205 ?"
150 GETA$:IFA$=""THEN150
160 IFA$="1"THENF=617:GOTO190
170 IFA$<>"2"THEN150
180 F=621
190 INPUT"WIEVIELE TOKEN";AT
200 IFAT>47THENPRINT"NICHT MOEGLICH":G
OTO190
205 REM *** BEFEHLSWORT ABSPEICHERN
210 Z=SA+F:W=SA+520
220 FORT=1TOAT
230 PRINT"TOKEN "T+204", ADRESSE"
240 INPUTA$,B:B=B-1
250 FORY=1TOLEN(A$)
260 X=ASC(MID$(A$,Y,1)):IFMID$(A$,Y+1,1)
=""THENX=X+128
270 POKEZ,X:Z=Z+1:NEXT
280 B2=INT(B/256):B1=B-B2*256
290 POKEW,B1:POKEW+1,B2:W=W+2
300 NEXT
310 DATA32,141,253

READY.
```

Adresse	Funktionsbeschreibung	Bemerkung	Übergaberegister
C613	Startadresse einer Programmzeile berechnen		Eingabe: Zeile in $14,15 Ausgabe: Adresse in $5F, 60
C807	Prüft auf Doppelpunkt	Zur Syntaxkontrolle	
CD8A	Ausdruck holen (z.B. Zeichen)	$D7FD wandelt Fließk. in Int.	
CAA0	PRINT-Befehl: Ausdruck oder String holen und ausdrucken		
CEF7	Prüft auf Klammer zu »)«	Syntaxkontrolle	
CEFA	Prüft auf Klammer auf »(«	Syntaxkontrolle	
CEFD	Prüft auf Komma	Syntaxkontrolle	
CF04	»SYNTAX ERROR« ausgeben		
D113	Prüft auf Buchstabe (A-Z)		
D248	»ILLEGAL QUANTITY« ausgeben		
D79B	Byte-Wert (0-255) holen	Bei der Argumentabfrage	

Tabelle 3. Einige der wichtigsten Routinen, mit denen man für Basic-Befehle Argumente, Satzzeichen etc. abfragen kann.

FOR T= 49310 TO 47565: PRINT CHR$(PEEK(T)):NEXT
ausdrucken, so sehen die Kommandos im Groß-/Kleinschrift-Modus folgendermaßen aus:
END = enD
FOR = foR ect.

Ferner ist jedem Basic-Befehl eine Adresse zugeordnet, bei der eine Abarbeitung vorgenommen wird. Diese Adressen sind in einer Tabelle (von $C00C $C07F) zusammengefaßt. Eine spezielle Routine hat wiederum die Aufgabe, die Befehlsadresse für ein entsprechendes Token aus der Tabelle zu lesen und einen Sprung nach dorthin durchzuführen. Auch dieses Unterprogramm kann man mittels eines Vektors umgehen.

Damit haben wir das nötige Rüstzeug, um selbst Basic-Kommandos in den Interpreter mit aufzunehmen. In Listing 4 ist ein dafür geeignetes Programm abgedruckt, welches ich jetzt näher erläutern möchte. Es ist der Kopf für ein Utility, in das nach Belieben Befehlswörter und Sprungadressen eingesetzt werden können.

Die Routine gliedert sich in vier Teile: Der erste Teil (von $2000 — $203A) ist eine Kopie der Reset-Routine. Dabei wird der Computer neu initialisiert und eine neue Kaltstartmeldung
**** 64'ER BASIC ****
ausgedruckt. Hier ist genügend Platz vorgesehen, damit der Text nach eigenen Wünschen gestaltet werden kann. Ferner wurden die ersten neun Bytes mit NOPs versehen, damit dort teils das Programm im Modulbereich abgelegt wird — die obligate Autostartinformation (a0CBM) eingesetzt werden kann. Anderenfalls — wenn das Maschinenprogramm per SYS gestartet wird — muß der erste Mnemonic-Befehl ein SEI sein.

Die sich jetzt anschließenden Programmteile sind die oben erwähnten Ergänzungen zu den Interpreterroutinen. Dies sind teilweise Kopien aus den alten ROM-Routinen mit Ergänzungen für die neuen Basic-Kommandos. Die Befehlswörter müssen jetzt nur noch eingetragen werden. Das geschieht folgendermaßen:

Die Befehle werden mit Hilfe des Programms »Tokenerzeuger« (Listing 5) in den entsprechenden Adreßbereich geschrieben. Bei nachträglichen Eintragungen ist zu beachten, daß zum letzten Buchstaben jedes Befehlswortes der Wert $80 (=128) zu addieren ist. Die Ergänzungsbefehle beginnen mit Token 204. Das erste Befehlswort beginnt mit dieser Nummer, der zweite erhält automatisch die 205 und so weiter.

Weiterhin ist für jedes Kommando die Sprungadresse in der Tabelle ($2208-$2268) zu vermerken. Dabei muß die Reihenfolge Low-/High Byte beachtet werden. Außerdem muß das LOW-Byte vor dem Eintrag um eines dekrementiert werden (LOW Byte -1). Auch diese Arbeit erledigt das Programm in Listing 5.

Ein Befehl wurde bereits eingetragen. Der Befehl KILL führt einen Reset durch und damit ist das Programm abgeschaltet. Wer also seine Maschinenprogramme ins Basic einbinden möchte, kann dies mit der beschriebenen Methode tun. Beispielsweise könnte man die in Listing 2 abgedruckte Funktionstastenroutine mit dem KEY-Befehl belegen. Auch im Hinblick auf Grafik, Tonerzeugung oder Joystickabfrage bietet sich hier die Möglichkeit, die Unzulänglichkeiten des Basics zu überwinden. Auch die Besitzer eines C 64 können die beschriebene Routine benutzen, da die Basic-ROMs nahezu identisch sind (sie haben lediglich eine andere Adresse). Außerdem habe ich in Tabelle 3 eine Liste der wichtigsten Unterprogramme zusammengestellt, mit denen man unter anderem Parameter, Strings, Kommas oder ähnliches abfragen kann. Auf jeden Fall sollte man sich für diese Arbeit ein ROM-Listing (zum Beispiel »VC 20 Intern« von Data Becker) zulegen.

Damit möchte ich für heute schließen. Das nächste Mal werden wir sehen, was man mit den Kernal-Vektoren anfangen kann und betrachten die Grafikmöglichkeiten beim VC 20.

(Christoph Sauer/ev)

Memory Map mit Wandervorschlägen

TEIL 1

Es steckt sehr viel im ersten Kilobyte des VC 20 und C 64. Wir werden Ihnen im Rahmen dieses Kurses die Bedeutung und Anwendung der Speicher und Register von Betriebssystem und Interpreter näherbringen.

Hinweise und Tips über nützliche PEEK- und POKE-Adressen gehören zum Standard-Repertoire einer Computer-Zeitschrift. Ebenso häufig werden Leserfragen zu diesem Thema gestellt, obwohl mehrere Handbücher für die beiden Home-Computer von Commodore bereits Speicherlisten (auf englisch »Memory Map«) enthalten.

Warum ich mich jetzt auch noch mit diesem Thema befassen will, hat zwei Gründe. Zum einen stört mich, daß ein Hinweis wie:

»...mit POKE 19,1 läßt sich das Fragezeichen bei INPUT-Befehlen unterdrücken...«

zwar richtig und auch anwendbar ist, aber halt nicht erklärt, was da eigentlich passiert und welche Folgen das für ein Programm haben kann. Zum anderen vermisse ich speziell in den Speicherlisten nähere, auch für den Anfänger verständliche und irgendwann einmal verwertbare Angaben.

Ich habe mir deshalb vorgenommen, Ihnen die Bedeutung und Anwendungen der PEEK- und POKE-baren Adressen, — sozusagen eine Wanderkarte mit Tourenvorschlägen und Sehenswürdigkeiten — in Form von Beispielen und Kochrezepten, näher zu bringen. Mir ist durchaus bewußt, daß das kein leichtes Unterfangen ist, da ich möglichst ohne Fach-Jargon auch für Nichttechniker verständlich bleiben möchte und da die Zahl der zu behandelnden Adressen recht hoch ist. Ich werde also um Kompromisse wohl manchmal nicht herumkommen. Bevor wir anfangen, möchte ich noch einen kleinen »Arbeitsplan« machen.

■ Zur Methode:
Meine Erklärungen sind so aufgebaut, daß sie am besten vor dem Computer mit der Zeitschrift auf den Knien nachvollziehbar sind, also »Lies und Tipp«.

■ Zum Adressenbereich:
Prinzipiell sind natürlich alle RAM-Adressen (RAM = Lese- und Schreibspeicher) POKEbar und kämen daher in Betracht. Vorerst aber werden wir uns nur den Bereich von 0 bis 1023 vornehmen.

■ Zum Computer:
Der genannte Speicherbereich hat mit wenigen Ausnahmen für VC 20 und C 64 die gleiche Bedeutung. Ich werde daher beide Computer gleichzeitig behandeln und auf Unterschiede jeweils gezielt hinweisen.

■ Der erste Hinweis:
In Tabelle 1 sind die Unterschiede in groben Umrissen zusammengefaßt.

■ Zur Darstellung:
Die Kenntnis der Bedeutung dieser Speicherzellen kommt auch Programmen in Maschinensprache zugute. Ich gebe daher alle Adressen sowohl als Dezimal- als auch als Hexadezimalzahl (mit vorgestelltem »$«) an.

■ Zu den Adressen:
Wenn in die zur Diskussion stehenden Speicherzellen eine Adresse aus dem erlaubten Bereich 0 bis 65535 ($0 bis $FFFF) hineingeschrieben wird, geschieht das immer mit der Aufteilung in einen niederwertigen Teil (Low Byte) und einen höherwertigen Teil (High Byte). Das Rezept zur Umrechnung finden Sie auf Seite 137.

Tabelle 1.
Unterschiede zwischen VC 20 und C 64

Adressen	Unterschied
0 — 2	sind verschieden
3 — 672	haben gleiche Funktionen
673 — 677	im VC 20 nicht benützt
678 — 767	in beiden nicht benützt
768 — 783	sind bei beiden gleich
784 — 787	im VC 20 nicht benützt
788 — 819	haben gleiche Funktionen
820 — 1023	sind bei beiden gleich

Wozu brauchen das Betriebssystem und der Basic-Übersetzer RAM-Speicherzellen?

Auf den ersten Blick ist nicht verständlich, warum die Speicherzellen von 0 bis 1023 feste Bedeutung haben und für normale Programme nicht zur Verfügung stehen. Wenn sie schon, wie es heißt, vom Betriebssystem und dem Übersetzer-Programm verwendet werden, warum stehen sie dann nicht gleich im ROM-Speicher bei allen anderen Teilen dieser Systeme?

Ein Computer führt einen Programmschritt nach dem anderen aus, ganz stur, ohne eigene Ent-

Memory Map

scheidungsfähigkeit, es sei denn, das Programm schreibt derartige Entscheidungen vor. Das Betriebssystem ist sozusagen im ROM eingefroren beziehungsweise festgeschrieben. Das würde aber bedeuten, daß der Computer keine Variationsmöglichkeiten hat, und daß alle Programme in gleicher Weise ablaufen. Aber das stimmt natürlich nicht! Alle Programme sind verschieden, sie belegen einen verschieden langen Speicherbereich und verarbeiten die unterschiedlichsten Variablen. Wir geben verschiedene Zeichen mit der Tastatur ein, der Computer wartet, bis eine Taste der Datasette gedrückt ist und so weiter.

Dafür braucht das Betriebssystem einen Speicherbereich, der variabel ist, in den es Zwischenwerte ablegen und später wieder auslesen kann.

Und das ist genau der Speicherbereich, der uns interessiert, nämlich von 0 bis 1023, womit wir wieder beim Thema wären.

Jetzt aber geht es los und zwar gleich in die Vollen. Denn ausgerechnet die ersten drei Speicherzellen haben laut Tabelle bei beiden Computern eine verschiedene Bedeutung und zusätzlich gehören sie mit zu den kompliziertesten.

Adresse 0 bis 2 ($0 — $2) beim VC 20: Sprungbefehl und wählbare »Sprungadresse« des USR-Befehls.

Die drei Adressen werden bei der Abwicklung des Basic-Befehls USR verwendet und stehen dem Programmierer zur Verfügung.

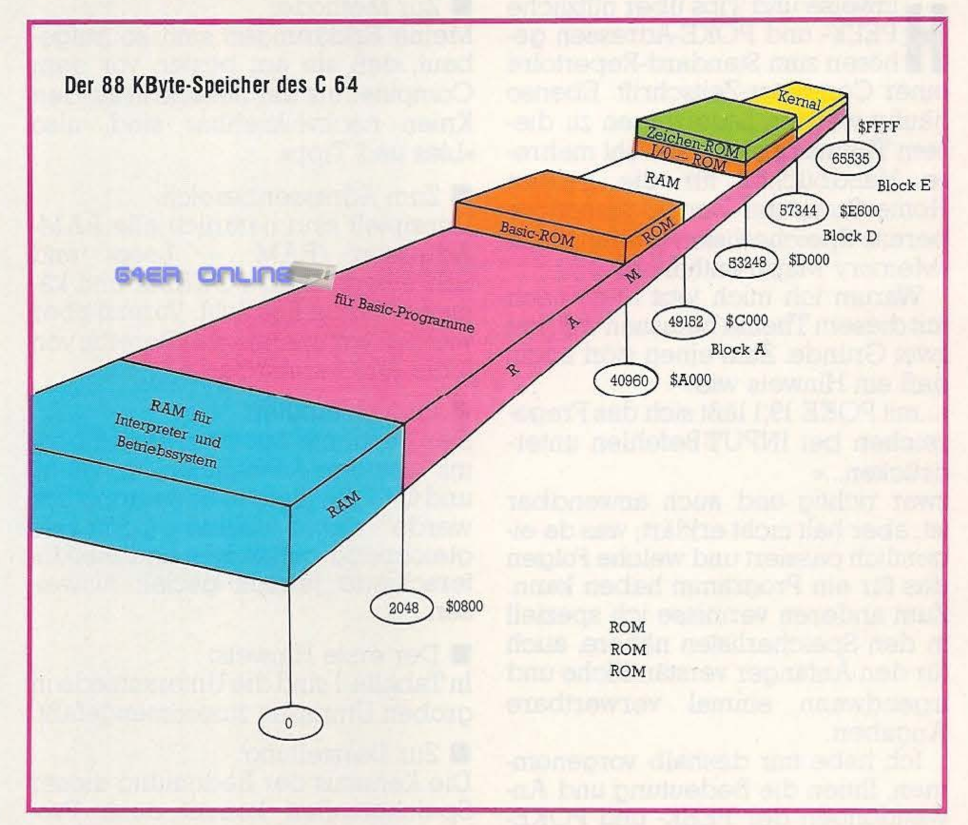
Hinweise: Diesen drei Adressen des VC 20 entsprechen beim C 64 die Adressen 784 ($310) bis 786 ($312). Die folgenden Erklärungen gelten also entsprechend auch für den C 64.

Hand aufs Herz: Haben Sie USR schon einmal benützt? Ohne Zweifel gehört dieser Befehl zu den seltenen. Ich will ihn daher hier kurz erläutern. USR hat dieselbe Funktion wie SYS, nämlich aus einem Basic-Programm direkt in ein Maschinenprogramm zu springen und dort solange weiterzufahren, bis mit dem Befehl RTS (entspricht dem Basic-Befehl RETURN) in das Basic-Programm zurückgesprungen wird. Die Sprungadresse in das Maschinenprogramm steht bei SYS gleich hinter dem Befehl.

Bei USR muß die Adresse zuerst in die Speicherzellen 1 und 2 (aha!!) gePOKEt werden.

Beispiel — Sprung auf 56524 ($DCCC):
mit SYS: SYS 56524
mit USR: POKE 1,204:POKE 2,220:X = USR(Y)

Kein Wunder, daß USR selten benützt wird. Aber erstens ist er durch das POKEn der Low-High-Byte-Darstellung aufgebläht und zweitens hat er auch wesentlich mehr Fähigkeiten als SYS.



Sein Argument, im obigen Beispiel also das »Y«, wird nämlich zuerst in den »Fließkomma-Akkumulator« FAC 1 (Floating Point Accumulator Nr. 1) gebracht, der sich in den Speicherzellen 97 bis 102 ($61 bis $66) befindet. Da wir ihn auf unserer Reise durch den Speicher noch treffen werden, brauche ich jetzt nicht näher darauf einzugehen. Wichtig ist lediglich, daß der Wert von »Y« dann vom angesprungenen Maschinenprogramm verarbeitet werden kann. Das Resultat kommt dann wieder in diesen FAC 1 und steht als Wert von X (siehe Beispiel oben) dem Basic-Programm zur Verfügung.

Mit USR kann man also Variable ins Maschinenprogramm zur Bearbeitung und zurück transferieren — und das ist der Unterschied zum SYS-Befehl. Ich möchte das an einem kleinen Beispiel demonstrieren. Statt allerdings ein Maschinenprogramm selbst zu schreiben, verwende ich, beziehungsweise springe ich, auf eine Routine des Betriebssystems, welches Werte des FAC 1 für mathematische Operationen verwendet.

Als mathematische Operation wähle ich das eingebaute Programm für INT, welches im VC 20 ab Speicherzelle 56524 ($DCCC) steht (im C 64 steht es ab 48332 ($BCCC)). Dieses wollen wir verwenden: In Zeile 10 definieren wir einen Wert für die Variable X, der in das Maschinenprogramm gebracht werden soll. Mit Zeile 20 bringen wir die Startadresse des Maschinenprogramms in die Speicherzellen 1 und 2.

Laut Kochrezept teilen wir die Adresse 56524 auf in ein Low-Byte = 204 und ein High-Byte = 220.

Der Befehl in Zeile 30 löst den gan-

mit Wandervorschlägen

zen USR-Vorgang aus, Zeile 40 gibt uns das Resultat.

```
10 Y=14.35
20 POKE 1,204:POKE 2,220
30 X=USR(Y)
40 PRINT X
```

Hinweis:
Entsprechend der anderen Adresse 48332 lautet die Zeile 20 beim C 64:

```
20 POKE 785,204:POKE 786,188
```

Nach RUN erhalten wir das Resultat 14, wie das Gesetz für INT es befiehlt. Natürlich hätten wir gleich PRINT INT (14.35) schreiben können, aber ich wollte ja nur demonstrieren. Der eigentliche Wert des USR-Befehls kommt hauptsächlich bei selbtgeschriebenen Maschinenprogrammen zum Zuge.

Sie können zur Übung im obigen Programm statt INT auch COS verwenden, indem Sie auf die Adresse 57935 ($E261) beziehungsweise beim C 64 auf 57938 ($E264) springen. Der Vergleich mit dem Basic-Befehl COS muß dasselbe Resultat ergeben.

Wer hat gemerkt, daß wir überhaupt nichts mit der Speicherzelle 0 gemacht haben, obwohl sie doch beim USR angeblich beteiligt ist?

Sie ist es wirklich, doch ohne unser Zutun. In diese Adresse wird beim Einschalten des Computers die Zahl 76 ($4C) geschrieben. Das ist der Code für den Maschinenbefehl »JMP«, der soviel bedeutet wie GOSUB. Bei USR springt nämlich das Programm auf die Speicherzelle 0, findet dort den Sprungbefehl und in den nachfolgenden Zellen 1 und 2 die Sprungadresse — und führt den Sprung auch gleich aus.

Jetzt aber wollen wir uns anschauen, wie diese drei Speicherzellen beim C 64 verwendet werden.

Adresse 0 ($0) beim C 64:
Datenrichtungsregister für Ein/Ausgabe-Port des 6510-Mikroprozessors

Adresse 1 ($1) beim C 64:
Datenregister für Ein/Ausgabe-Port des 6510 — Mikroprozessors,

Adresse 2 ($2) beim C 64:
unbenutzt

Im Gegensatz zum Mikroprozessor des VC 20 hat der des C 64 sechs Ein/Ausgabe-Leitungen die einzeln programmierbar sind und so eine direkte Verbindung zwischen dem Mikroprozessor und der Außenwelt herstellen. Warum nur sechs Leitungen und nicht wie üblich acht? Auf dem Chip selbst könnten acht Bit verkraftet werden, aber es stehen nur sechs Anschlußbeine zur Verfügung.

Um trotzdem flexibel zu bleiben, ist dieses Tor zum Prozessor — zutreffend auch »Port« genannt — in beiden Richtungen begehbar. Jede einzelne der sechs Leitungen kann vom Programmierer auf »Eingang« oder auf »Ausgang« geschaltet werden. Dazu dient das Datenrichtungsregister in der Speicherzelle 0.

Datenrichtungs-Register in Zelle 0

Wenn zum Beispiel in das Bit 4 der Zelle 0 eine 0 hineingePOKEt wird, ist die Leitung Nummer 4 des Ports auf »Eingang« geschaltet. Es gilt für alle 6 Bits (Nummer 0 bis 5):
— Bit auf 0 = Eingang
— Bit auf 1 = Ausgang

Beim Einschalten schreibt das Betriebssystem in dieses Register die Dualzahl ..101111 (dezimal=47). Das heißt also, daß nur die Leitung Nummer 4 als Eingang verwendet wird, alle anderen aber als Ausgang. Warum das so ist, sehen wir gleich. Vorher will ich aber noch erwähnen, daß im C 64 von dieser Flexibilität des Mikroprozessor-Ports kein Gebrauch gemacht wird. Ich habe das ganze Betriebssystem durchgesehen, aber das einzige Mal, wo die Speicherzelle 0 angesprochen wird, ist eben bei der Einschaltroutine. Das heißt aber nicht, daß Sie, lieber Hobby-Programmierer, darauf verzichten müssen. Ich kann mir vorstellen, daß besonders Ausgefuchste unter Ihnen durch POKEn eines anderen Bitmusters in die Speicherzelle 0 vielseitige Befehle erzeugen und einsetzen können.

Das wird besonders deutlich, wenn Sie jetzt sehen, mit welchen Teilen des Computers diese sechs Leitungen verbunden sind.

Datenregister in Speicherzelle 1

Mit diesem Register steuert der Mikroprozessor (und damit natürlich das Betriebssystem) die Auswahl von Speicherblöcken und den Betrieb mit dem Kassettenrecorder. Dem Programmierer steht diese Möglichkeit über POKEn auch zur Verfügung.

Bit 0
schaltet den Speicherbereich 40960 — 49151 ($A000 — $BFFF) zwischen dem Basic-Übersetzer (Interpreter) im ROM und freiem RAM um (Normalzustand=1)

Bit 1
schaltet den Speicherbereich 57344 — 65535 ($E000 — $FFFF) zwischen dem Betriebssystem (Kernal) im ROM und freiem RAM um (Normalzustand=1)

Bit 2
schaltet den Speicherbereich 53248 — 57343 ($D000 — $DFFF) zwischen Zeichen-ROM und Ein/Ausgabe-ROM um (Normalzustand =1)

Bit 3
sendet serielle Daten zum Kassettenrecorder (Normalzustand=0)

Bit 4
prüft, ob eine der Tasten des Recorders gedrückt ist, welche den Motor einschalten (Normalzustand =1)

Bit 5
schaltet den Motor des Recorders ein und aus (Normalzustand 1)

Als erstes möchte ich die RAM-ROM-Umschaltung näher beschreiben.

Sie wissen, daß Ihr C 64 deswegen so heißt, weil er 64 KByte Speicherplätze hat. Nur stimmt das nicht! Er hat nämlich 88 KByte und müßte eigentlich C 88 heißen.

Da mit den 16 Bit der High/Low-Byte Methode (siehe Bild 1) nur 64 KByte adressierbar sind, müssen die restlichen 22KByte bei Bedarf eingeschoben werden — und das machen die oben erwähnten Bits 0 — 2 des Datenregisters.

In Bild 2 sehen Sie die drei oben erwähnten Speicherblöcke, die sowohl mit RAM als auch mit ROM belegt sind, einer davon gleich doppelt. Ich habe ihnen folgende Namen gegeben:
— 40960-49151 ($A000-$BFFF) = BLOCK A
— 53248-57343 ($D000-$DFFF) = BLOCK D
— 57344-65535 ($E000-$FFFF) = BLOCK E

Tabelle 2 gibt Ihnen die Übersicht über die gemeinsame Wirkung der Bits 0, 1 und 2 des Datenregisters auf den jeweiligen Inhalt der Speicherblöcke.

Memory Map

# 2	1	0	BLOCK A	BLOCK D	BLOCK E
1	1	1	Basic	I/0	Kernal
1	1	0	RAM	I/0	Kernal
1	0	1	RAM	I/0	RAM
1	0	0	RAM	RAM	RAM
0	1	1	Basic	Zeichen	Kernal
0	1	0	RAM	Zeichen	Kernal
0	0	1	RAM	Zeichen	RAM
0	0	0	RAM	RAM	RAM

Dabei bedeuten
- Basic: Basic-Übersetzer (Interpreter)
- I/0: Ein/Ausgabe-Register
- Zeichen: Zeichenspeicher
- Kernal: Betriebssystem
- RAM: frei verfügbarer Speicher

Tabelle 2. So sind die Bits 0, 1 und 2 des Datenregisters mit dem Inhalt der Blöcke A, D und E verknüpft.

Wie Sie durch PRINT PEEK (1) selbst leicht feststellen, steht nach dem Einschalten des Computers im Register 1 die Zahl 55. In dualer Darstellung ist das 110111. Das entspricht dem oben genannten »Normalzustand« der einzelnen Bits.

Vergleichen Sie es bitte mit der Auflistung am Anfang der Beschreibung der Speicherzelle 1. Die in Tabelle 2 dargestellten Bits sind also die rechten drei Bits der Zelle 1. Lassen wir die Bits 3, 4 und 5 unverändert, ergeben die acht Kombinationen der Tabelle 2 die Zahlen 55 bis 48. Durch den Befehl POKE 1,54 können wir nun den Basic-Übersetzer ausschalten und 8 KByte Speicher gewinnen. Nur nutzt uns das nicht viel, denn was tun — ohne Basic! Es gibt aber doch eine Anwendung. Zuvor will ich Ihnen aber noch beweisen, daß wir tatsächlich den Block A auf RAM umschalten. Der Trick besteht darin, den Basic-Übersetzer vom ROM in den darunter liegenden RAM umzuladen. Wenn er tatsächlich in RAM steht, müßten wir ihn durch POKEn verändern können zu einem Privat-Basic. Geben Sie direkt ein:

FOR J=40960 TO 49151: POKE J, PEEK(J):
NEXT J

POKE J,PEEK(J) — das sieht dümmer aus als es ist. Die »Doppeldecker-Speicher« erlauben nämlich ein PEEKen nur aus dem ROM-Bereich. Ein hineinPOKEn dagegen geht nur in den RAM-Teil. Von dort aber kann er — wie gerade gesagt — nicht herausgelesen werden, es sei denn, wir schalten um!

Merken Sie was? Die Zeile oben liest also den Inhalt des Basic-ROMs und schreibt ihn in den RAM mit identischen Adressen. Die Ausführung der Zeile braucht einige Zeit. Wenn der Cursor wieder blinkt, schalten wir den RAM ein mit:

POKE 1,54

Wir merken natürlich noch keinen Unterschied, denn das RAM-Basic ist ja noch dasselbe, wie es im ROM steht.

Doch nun werden wir es verändern. In der Speicherzelle 41220 steht das »P« für den Befehl PRINT mit dem ASCII-Codewert 80. Dieses P ersetzen wir durch ein »G« (ASCII-Code = 71).

POKE 41220,71

Versuchen Sie bitte, mit dem (nicht durch »?« abgekürzten) PRINT-Befehl ein Zeichen auf den Bildschirm zu drucken. Es wird Ihnen nicht gelingen, denn der Befehl heißt jetzt:

GRINT "A"

was beweist, daß das Basic jetzt in RAM steht. Das Umdefinieren von Befehlen ist natürlich wenig sinnvoll. Aber wer die Maschinenprogramme des Basic kennt, kann sie auf diese Weise ändern, erweitern, einschränken, solange er sich auf in sich geschlossene Teile beschränkt.

Eine inzwischen oft zitierte Anwendung stammt von Jim Butterfield (siehe Literatur), den es begreiflicherweise stört, daß der Befehl ASC, welcher den ASCII-Code eines Strings erzeugt, bei einem Null-String das Programm mit ILLEGAL QUANTITY ERROR beendet. Versuchen Sie es:

PRINT ASC ("A") ergibt die Zahl 65.
PRINT ASC ("") hat die obige Fehlermeldung zur Folge.

Wenn Basic im RAM steht, können wir das ändern:

POKE 46991,5

Die Wiederholung des Befehls PRINT ASC ("") ergibt jetzt 0 — und, was das Wichtige ist, das Programm läuft weiter.

Durch zusätzliches Umladen des Speicherblocks E und anschließendes Umschalten mit POKE1,53 ist auch das Betriebssystem veränderbar — ein weites Feld für fortgeschrittene Programmierer in Maschinensprache.

Die wohl wichtigste Anwendung der Umschaltmethode wird den Maschinen-Programmierern geboten, die dadurch eine kostenlose Speichererweiterung von 16 KByte erhalten. Bei gleichzeitiger Verwendung von Basic und Maschinenprogramm kann die Umschaltung besonders vorteilhaft eingesetzt werden. Das Umschaltprogramm muß dann aber ebenfalls in Maschinensprache geschrieben sein und darf nicht im Umschaltbereich liegen.

Das Umschalten von den Ein/Ausgabe-Registern des Blocks D mit POKE 1,51 erlaubt, die Bitmuster der fest programmierten Zeichen aus dem Zeichen-ROM auszulesen, in einen freien RAM-Bereich zu bringen und dort dann nach eigenen Vorstellungen zu verändern. Im Grafik-Kurs war das ausführlich beschrieben.

Der Vollständigkeit halber muß ich hier noch erwähnen, daß neben den drei ersten Bits der Speicherzelle 1 noch zwei weitere Signale die RAM/ROM-Umschaltung beeinflussen. Es sind das die Leitungen auf Pin 8 und 9 des Erweiterungssteckers (GAME und EXROM), welche durch Spiel- und Programmodule benützt werden. Eine genaue Beschreibung der dadurch erzeugten sinnvollen Speicherkombinationen finden Sie in dem Buch »64 Intern« von Data Becker ab Seite 14.

Bit 3, 4 und 5 regeln wie schon gesagt den Betrieb des Kassettenrecorders.

Zu **Bit 3** ist oben schon alles notwendige gesagt.

Bit 4 ist im Normalzustand auf 1, »normal« heißt hier, solange keine der

mit Wandervorschlägen

Motor-Tasten der Datasette (PLAY, REWIND, FAST FORWARD) gedrückt ist.
Zur Probe:

```
10 X=PEEK(1)
20 PRINT X
40 GOTO 10
```

Betrieb des Kassettenrecorders

Die schon erwähnte »Normalzahl« 55 (dual = 110111) läuft als Zahlenband solange, bis eine der besagten Tasten gedrückt wird. Dann läuft eine 7 (dual = 000111). Warum auch Bit 5 zu 0 wird, kommt gleich nachher zur Sprache.

Mit einer kleinen Erweiterung der drei Zeilen können Sie in einem Programm den Status der Motor-Tasten abfragen. Ergänzen Sie:

```
30 IF X=7 THEN 50
50 PRINT »TASTE GEDRÜCKT«
```

Um nur Bit 5 abzufragen, schreiben wir besser:

```
30 IF (X AND 16)=0 THEN 50
```

Diese Abfrage kann allerdings nicht unterscheiden, welche der drei Tasten der Datasette gedrückt worden ist. Außerdem funktioniert das alles nur, wenn — wie im »Normalfall« — das Bit 4 des Datenrichtungsregister (Speicherzelle 0) auf 0 (Eingang) steht.

Bit 5 schaltet den Motor der Datasette ein und aus. Es bietet sich an, damit per Programm die Datasette zu schalten — wenn so etwas nützlich ist. Leider ist dieses Bit etwas schwieriger zu handhaben, da es in der Interrupt-Routine des Betriebssystems eine Rolle spielt.

Die Tasten der Datasette werden nämlich 60mal in der Sekunde abgefragt. Wenn keine Taste gedrückt ist, setzt das Betriebssystem sowohl das sogenannte »Interlock«-Register in Speicherzelle 192 auf 0 als auch Bit 5 der Zelle 1 auf 1, wodurch der Motor ausgeschaltet wird beziehungsweise bleibt. Da kann man nicht dagegen an. Wir haben nur eine Chance, wenn eine Taste bereits gedrückt ist und der Kassettenmotor schon läuft.

Dann nämlich können wir zuerst das Interlock-Register mit einem Wert größer als 0 lahmlegen:
POKE 192,255
Jetzt läßt sich der Motor der Datasette mit Bit 5 steuern:
POKE 1,39 beziehungsweise
POKE 1,PEEK(1) OR 32
schaltet den Motor aus,
POKE 1,7 beziehungsweise
POKE 1,PEEK(1) AND 31
schaltet den Motor ein.

Das Interlock-Register in Speicherzelle 192 werde ich später noch einmal erwähnen, da es seine Funktion auch beim VC 20 ausübt, allerdings mit anderen Ein/Ausgangs-Ports. Das ist alles, was zur Speicherzelle 1 zu sagen ist. Das nächste Mal wird unsere Speicherreise weitergehen, wobei natürlich nicht alle Speicherzellen soviel hergeben beziehungsweise dem Programmierer so direkt zur Verfügung stehen, wie die Adressen 0 und 1.

(Dr. Helmuth Hauck/aa)

Literatur:
1) H. Ponnath, »Reise durch das Wunderland der Grafik«, 64'er, Ausgabe 4/84 und folgende
2) J.Butterfield, »The Architecture of the 64«, Commodore User, Sept. 1983
3) M.Angerhausen et al., VC 20 Intern, Data Becker, 1983
4) M.Angerhausen et al., 64 Intern, S. 11-21 Data Becker, 1983
5) S.Leemon, Mapping the C 64, COMPUTE Publications, 1984

Die Low-High-Byte-Methode

DARSTELLUNG VON ADRESSEN GRÖSSER 255

Um große Zahlen darzustellen, wird von allen Homecomputern die sogenannte High-Low-Byte-Methode angewendet.

Eine Speicherzelle der Commodore Computer ist 8 Bit lang, das ist 1 Byte, und sie kann daher als größte Zahl 255 ($FF) enthalten. Für Zahlen größer als 255 hängen wir mehrere Speicherzellen hintereinander, in unserem Fall deren zwei. Mit 2 Bytes (16 Bit) können wir nämlich maximal 65535 ($FFFF) darstellen. Diese Aufspaltung einer Adresse in zwei Speicherzellen soll das folgende Beispiel verdeutlichen.

dezimal		**47491**		
dual	1011	1001	1000	0011
hex $	B	9	8	3
High-Byte	185			
Low-Byte			131	

Sie sehen, daß die Dezimalzahl 47491 aufgespalten wird in $B9 = 185 und $83 = 131. Zur Erinnerung: Jede Stelle einer Hex-Zahl kann direkt in eine 4stellige Dualzahl und umgekehrt gewandelt werden (1011 = $B, 1000 = $8).

Der Umrechnungsweg über eine 16stellige Dualzahl ist natürlich viel zu aufwendig. Ich empfehle Ihnen folgendes Kochrezept:
(1) Dezimal High/Low-Byte:
47491 : 256 = 185, Rest 131
Der Rest fällt bei der Division per Hand automatisch an. Mit dem (Taschen-)Rechner erhält man den Rest durch:
185 * 256 — 47491 = -131
2) High/Low-Byte Dezimal:
HB* 256 + LB = Dezimal
185* 256 + 131 = 47491
Wichtige Regel:
Die Mikroprozessoren von VC 20 und C 64 verlangen, daß immer das Low-Byte vor dem High-Byte kommen muß. Die Zahl wird sozusagen von rechts nach links gelesen (im Beispiel: 131, 185)







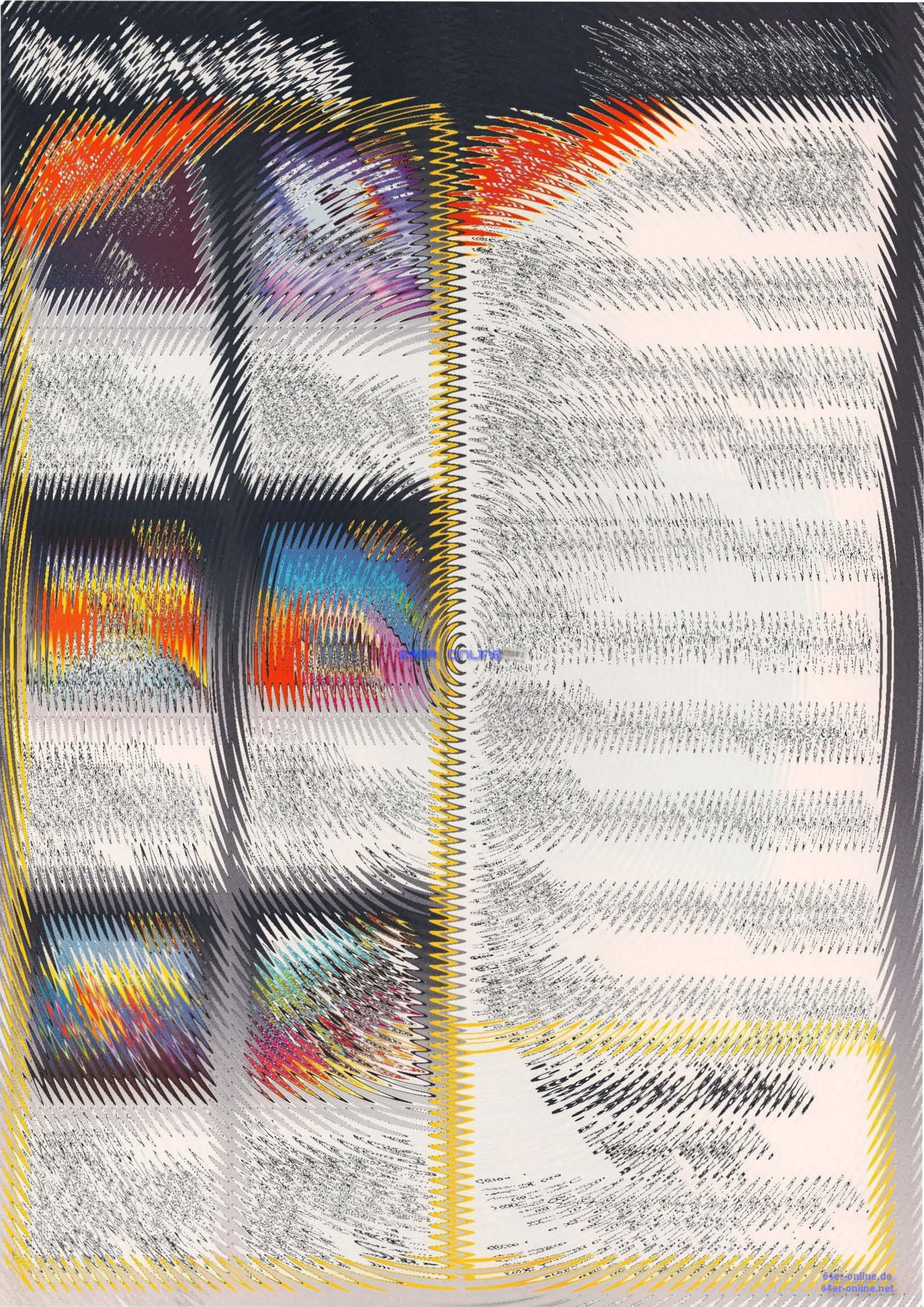




Bild 3. Studioatmosphäre.
Die realfotografische Seite der Modeanzeige.

Bild 1. Erster Tappser in Richtung Computergrafik war ein Single-Cover, erstellt mit einem TRS-80.

Aus anfänglichem Hobby wurde ernsthafte Anwendung. In seinem Münchener Hinterhofstudio verbindet der Fotograf Dirk Franke Schönheiten und Mode mit auf einem C 64 erzeugter Computer-Grafik. Die Ergebnisse können sich sehen lassen.

Das Lehel ist eines der ältesten und gemütlichsten Münchner Stadtteile. Hier wohnen neben Alteingesessenen Ausgeflippte aus den Bereichen Mode, Musik und Film. Versicherungskonzerne residieren protzig neben Tante-Emma-Läden, und vor nicht allzu langer Zeit gab es in diesem Viertel noch Münchens einzigen Pferdemetzger und sogar einen letzten, privaten Bauernhof. Auch heute gibt es dort noch viele kleine Handwerksbetriebe und Hinterhofwerkstätten. Und in einer von diesen beginnt unsere Geschichte.

In den Räumen einer aufgelassenen Galvanisierfabrik entstand mit viel Eigenleistung und Umbauarbeit das Lime-Light Studio. Dort machte sich nach langen »Lehrjahren« bei Profis der Mode- und Werbefotograf Dirk Franke mit einem eigenen Studio selbständig. Und lange sah es auch so aus, als würde dieses Studio ein Fotostudio wie so viele andere bleiben. Wäre da nicht...



Bild 2. Dirk Franke an seinem Arbeitsplatz. Links ist die Hardcopy der Computer-Grafik zu erkennen.

Bild 4. Das fertige »Produkt«, wie es als Anzeige in einer Mode- und Kulturzeitschrift erschienen ist.

Tja, wäre da nicht jemand aus dem großen Bekanntenkreis des jungen Fotografen mit einem TRS-80-Computer samt Monitor eines Tages im Studio bei Dirk Franke aufgetaucht.

Anfangs diente dieses heute schon antiquare Gerät nur zum Daddeln. Wer die eher einfachen bis langweiligen Spiele aus dieser Ära noch kennt, kann sich leicht vorstellen, daß bald mehr gefragt war. So machte Dirk Franke seine ersten Tapser in Basic. Und obwohl die grafischen Fähigkeiten dieses Modells noch nicht einmal bescheiden zu nennen waren, faszinierte ihn die Möglichkeit, mit einem solchen Gerät Bilder, wenn auch sehr primitive, erzeugen zu können.

Anders als die mit seiner Fotokamera geschossenen, hatten diese Bilder von sich aus schon ein gewisses Eigenleben, das man mit der herkömmlichen Fotografie erst mühsam und kreativ erzeugen mußte. Einen ersten Gehversuch, dieses für den Fotografen völlig neue Medium auch beruflich zu nutzen, zeigt das Cover für die Single »Letric Metric« von Peter Griffin (Bild 1), für den er das Titelbild vor nun fast drei Jahren knipste.

Der TRS-80 verließ einige Tage später samt dem Bekannten wieder das Studio. Aber von nun an war es klar, ein Computer mußte her. Es vergingen zwar noch ein paar Monate, doch dann war der C 64 und mit ihm bis dahin in dieser Preisklasse ungekannte grafische Möglichkeiten auf dem Markt.

Dennoch, mit der schnellen Entscheidung für diesen Computer war es nicht getan. Jeder kennt die umständliche Programmierung von Grafiken beim C 64, ohne Tools oder Grafikerweiterungen. Es folgten also die Floppy und das Koala Pad. Jetzt konnte es endlich losgehen. Denkste!

Nach nächtelangen Versuchen und endlosem Testen mit den verschiedensten Kameratypen, unterschiedlichen Filmmaterialien, ja sogar variierenden Film-Emulsionen, die einzelne Farben nuanciert anders wiedergaben, und nachdem Dirk Franke sämtliche Laboranten

Mode Fotos mit Bits und Bytes

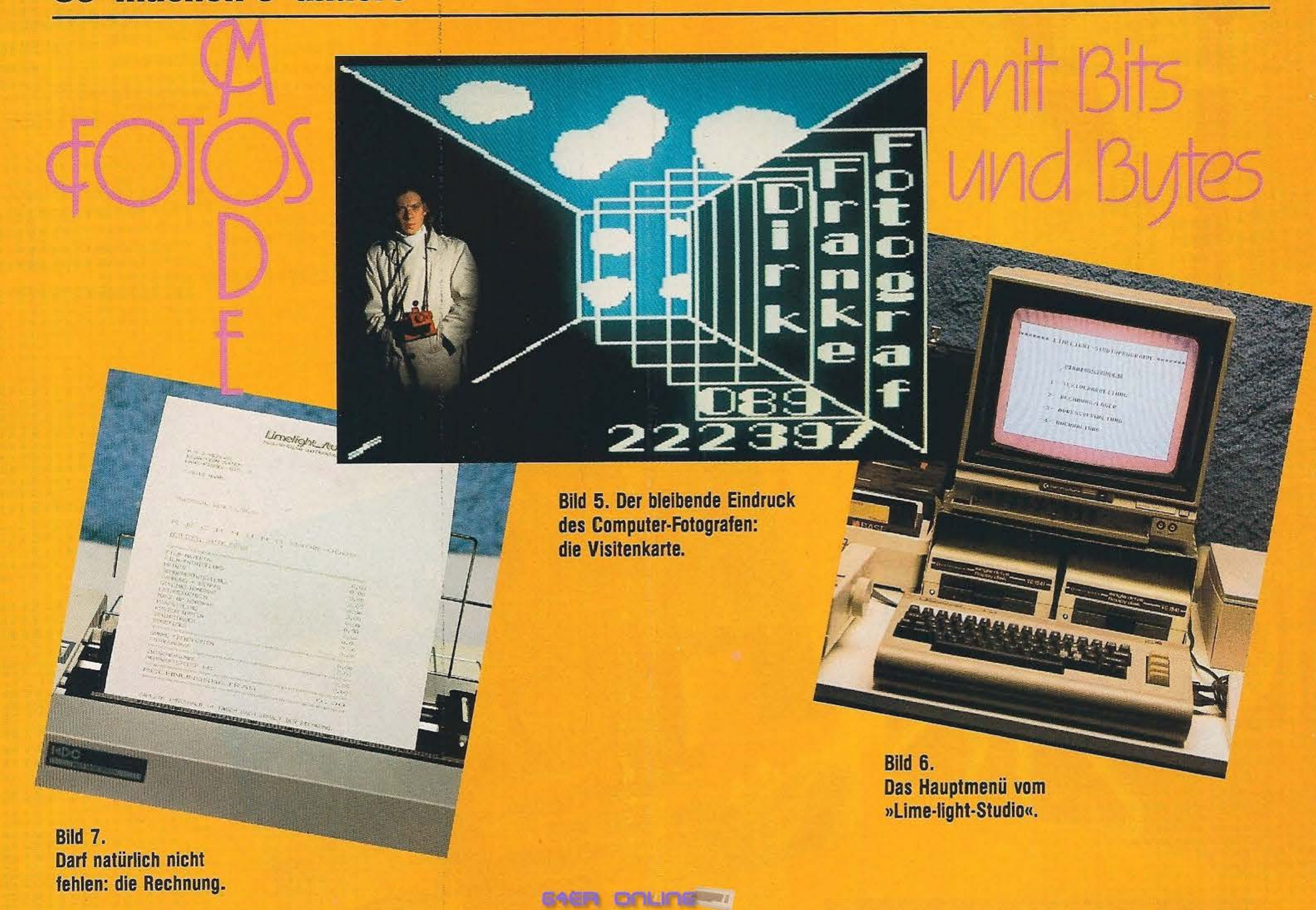


Bild 5. Der bleibende Eindruck des Computer-Fotografen: die Visitenkarte.

Bild 6. Das Hauptmenü vom »Lime-light-Studio«.

Bild 7. Darf natürlich nicht fehlen: die Rechnung.

im Entwicklungslabor kirre gemacht hatte, wollte er beinahe aufgeben. Die Qualität war einfach nicht akzeptabel.

Wieder mal ging es quer durch den Dschungel des Münchner Computer-Handels, bis ein geeigneter Farbmonitor gefunden war. Mit diesem ließ sich dann auch die letzte Hürde, die vorher viel zu grobe Auflösung, nehmen. Die neuen Tests waren vielversprechend, und ein erster Kunde zeigte Mut und Interesse an der neuen »Technik«.

Eine junge Modefirma suchte nach etwas Neuem: einem »Eyecatcher«. Denn nach wie vor gilt auch in dieser Branche die Devise des Auffallens um jeden Preis. Und dafür schien die Frankesche Verbindung von Real-Fotografie und Computergrafik wie geschaffen.

Mittels des Tabletts vom Koala Pad und der mitgelieferten Schrift-Software entstand das Computerbild (Bild 2). Ein Modell war schnell gebucht und stellte sich mitsamt der, damit sie nicht reflektieren, an speziell behandelten Nylonschnüren aufgehängten Jeans vor den schwarzen Hintergrund in Pose (Bild 3).

Stimmte erst einmal das Licht und stand die Grafik, war der Rest nur noch für den Assistenten Plackerei. Er war es nun, der die Filmkassetten für die ständigen Mehrfachbelichtungen quer durchs Studio hin- und herbringen mußte. Zweimal Klick mit Modell und Jeans. Blitze aus. Zweimal Klack am Computer. Blitze wieder an. Und immer weiter so, mindestens hundertmal.

Das Endergebnis war dann die Vorlage für die Anzeige und der Modekunde zufrieden. Doch schon während der Aufnahmen hatte der Layouter der Werbeagentur Gelegenheit zu letzten Korrekturen. Ein kurzes Basic-Programm ermöglicht eine Hardcopy des Bildschirms auf dem grafikfähigen Drucker. So konnte der Art-Director immer wieder Varianten scribbeln oder Veränderungen am Bildaufbau vornehmen, bevor der letzte Schuß im Kasten und die Aufnahmen damit gestorben waren (Bild 4).

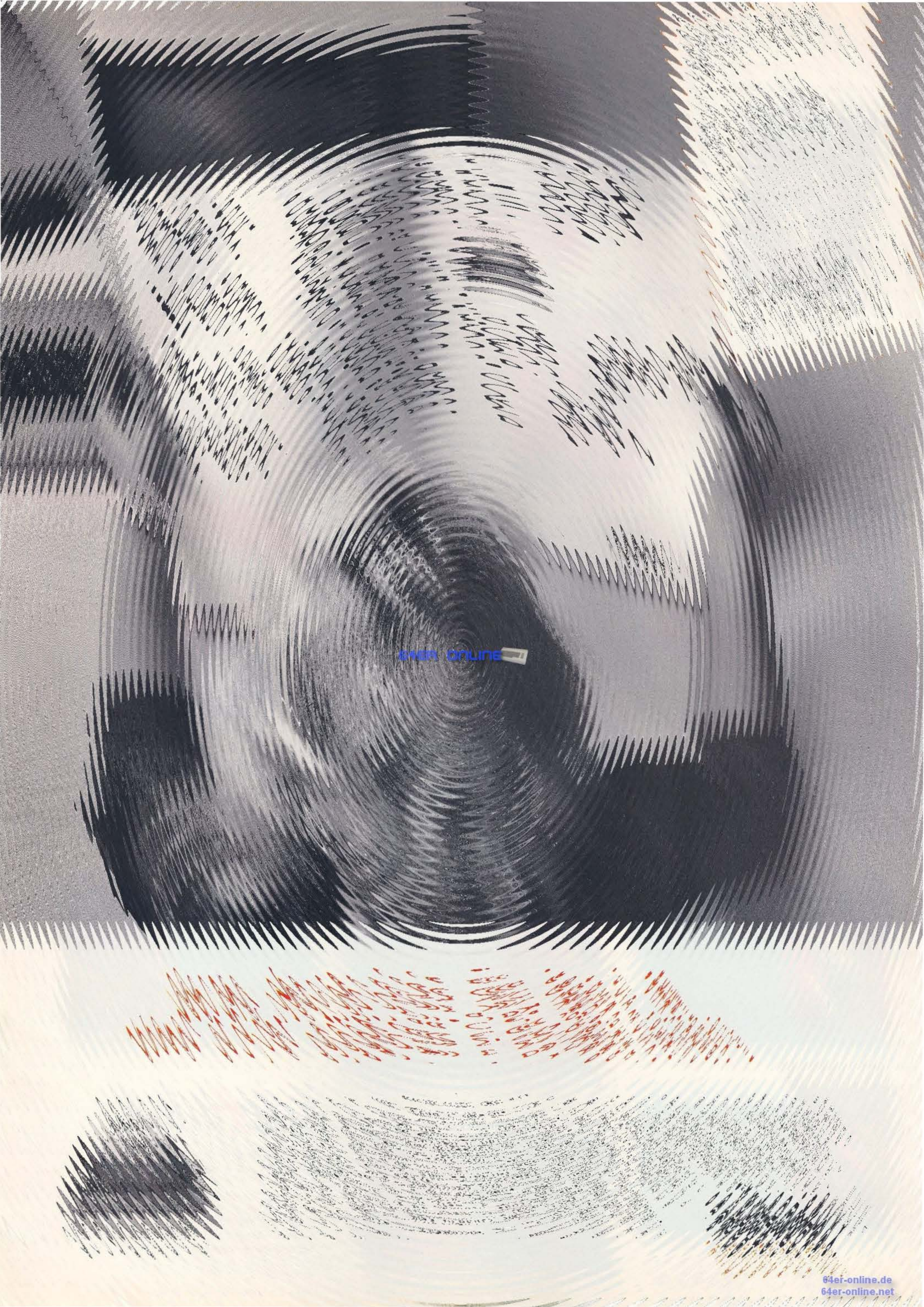
Welche Faszination diese Verbindung zweier, eigentlich artfremder Medien auf den Fotografen Dirk Franke ausübt, zeigt auch seine Visitenkarte: Ein 6 x 6-Dia, das den realfotografierten Meister im Bogie-Look zusammen mit Computerbild und Telefonnummer, einen bleibenden Eindruck hinterlassend zeigt (Bild 5).

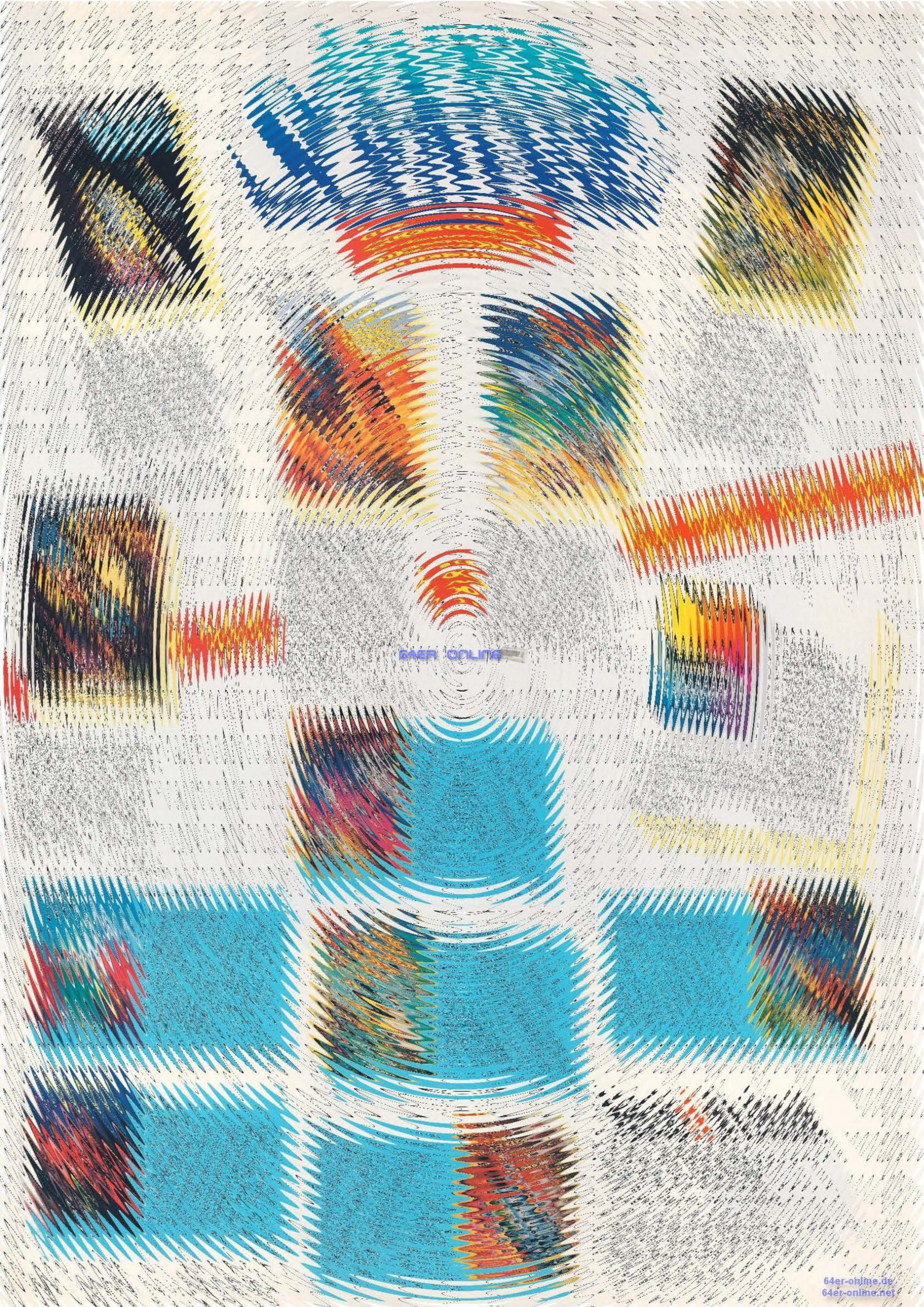
Doch nicht nur das fotografische Leben hat der Computer bei Dirk Franke nachhaltig beeinflußt. Auch in die interne Organisation seines »Ladens« brachte der C 64 neuen Schwung. Der ganze Papierkram befindet sich nun auf Disketten und teils selbstgeschriebene, teils gekaufte Programme erleichtern und übernehmen das Halten der täglichen Ordnung.

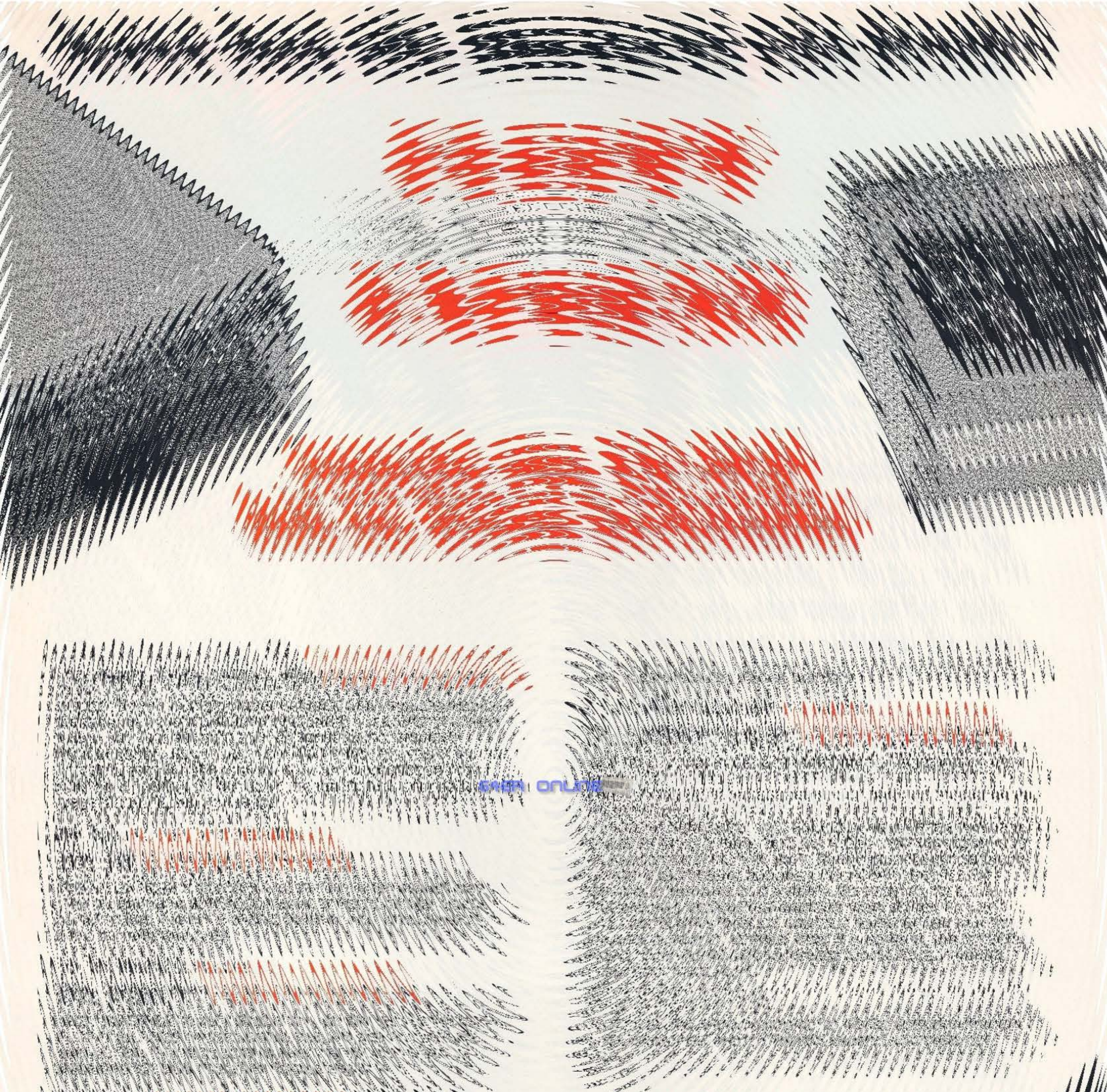
Vom Hauptmenü (Bild 6 und 7) geht es zur Textverarbeitung, zum Rechnungsstellung- und Film- und Requisitenlager-Programm, in die Einnahmenüberschuß-Buchhaltung und in die Adressenverwaltung, die für die zahlreichen, berühmtberüchtigten Studiofeste besonders wichtig ist.

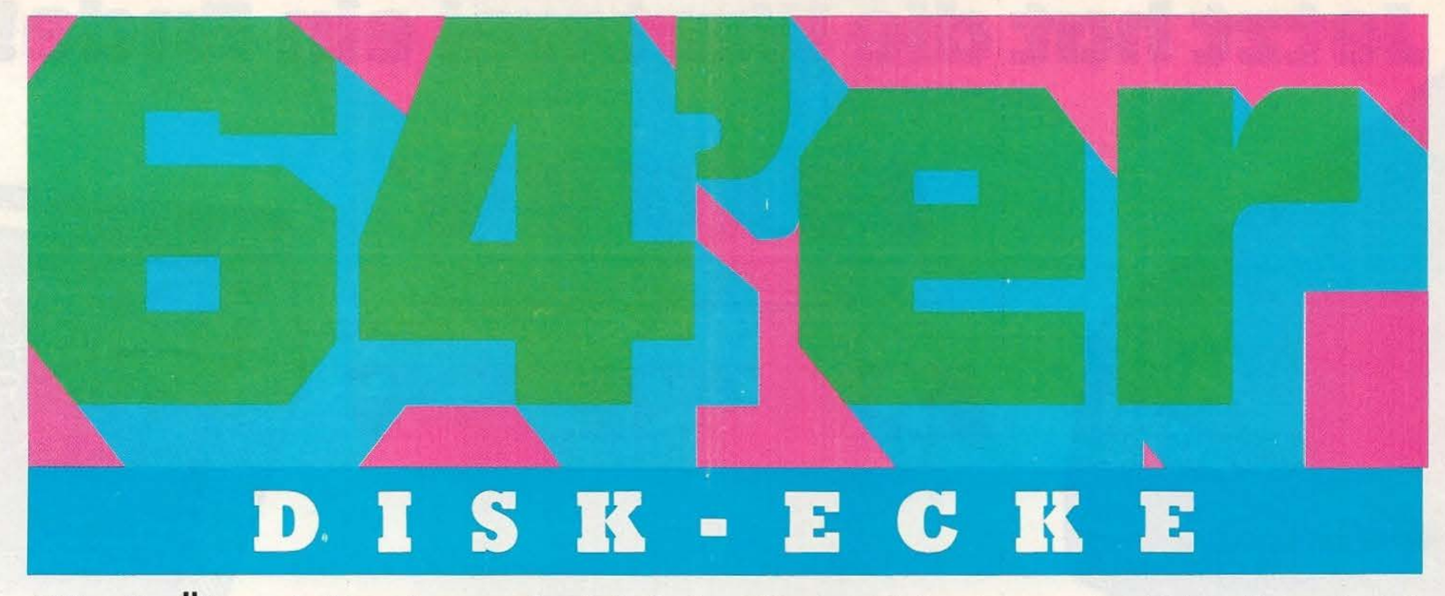
Dirk Franke möchte jedenfalls seinen »Compi« nicht mehr missen. Und seit es Spiele wie den Flugsimulator gibt, daddelt er auch manche Nacht mal wieder. Doch obwohl bereits erste Anzeigen mit seinen Computerbildern erschienen sind und er eigentlich recht zufrieden sein könnte, kommen angesichts der zahlreichen Veröffentlichungen über die Möglichkeiten großer Grafikcomputer bereits neue Wünsche auf. Verständlich...

(Klaus Koch/aa)









Wie die Überschrift schon andeutet, hat sich eine Änderung vollzogen. Das »Ka« für Kassette ist weggefallen. Dafür hat sich »Di« zu Disk gemausert. Es ist uns also endlich gelungen, die Programme auch auf Diskette anzubieten. Wir mußten allerdings eine Entscheidung fällen:

Kassette oder Diskette, beides ging nicht. Die Diskette ist aufgrund ihrer Verbreitung ausgewählt worden. Dafür sind jetzt alle Programme einer Ausgabe (VC 20 und C 64) auf einer Diskette erhältlich.

Eines hat sich aber nicht geändert: der Preis. Die Diskette für eine Ausgabe kostet demnach 29,90 Mark. Sie werden bei einigen Disketten bestimmte Programme vermissen. Deren Autoren konnten sich nicht entschließen, ihr Programm im Rahmen des Leserservice für eine Verbreitung auf Datenträger freizugeben. Bei den Ausgaben 4, 5 und 6 können noch Kassetten (VC ...) bestellt werden. Auf kurze Programme wurde aus Gründen der Übersichtlichkeit verzichtet. Nun noch einige technische Details. Zu den Programmen sind immer die Seitenzahlen angegeben, unter der Sie die Beschreibungen in der entsprechenden Ausgabe finden können. Der Diskette liegen also keinerlei Informationen bei. Lesen Sie daher aufmerksam die Anleitung (ob SYS-Befehle nötig sind, in welcher Reihenfolge geladen werden muß, eventuelle Sprach- oder Speichererweiterungen und ähnliches mehr) in dem jeweiligen Artikel nach. Aus Aktualitätsgründen wird jeweils die abgedruckte Version angeboten. Eventuelle systematische Fehler, die sich noch im Programm befinden können, müssen von Ihnen selbst, nach Studium des Druckfehlerteufelchens, korrigiert werden.

Fehlende Hefte erhalten Sie bei: Markt & Technik Vertrieb 64'er Hans-Pinsel-Str. 2, 8013 Haar

Ausgabe 11/84

Bestellnummer **CB 020**

Commodore 64

Turtle Grafik (LdM)	S.48
Schachmeister (AdM)	S.50
SMON (1. Teil)	S.59
Floppykurs	S.117
FPLOT-Befehlserweiterung	S.73
Get Koala pic	S.66
Interrupttechnik	S.84
Exsort (UPB)	S.154
Einzeiler	S.158
Simons Basic Befehlserweiterung (SB)	S.90

VC 20

Pseudosprites (8K)	S.76
Laterna Magica (8K)	S.68
Betriebssystem-Erweiterung (24K>)	S.88
Supergrafik (GV)	S.71
VC 20-Kurs (GV>)	S.126

Ausgabe 10/84

Bestellnummer **CB 019**

Commodore 64

Finanzmathematik (AdM)	S.68
Hypra-Load (LdM)	S.67
Hardcopy Compact 2	S.86
Hardcopy MPS 801	S.82
Hardcopy VC 1526 neu	S.83
Hardcopy Gemini-10X	S.85
Hardcopy FX-80	S.88
Hardcopy VC 1520 farbig	S.84
Apocalypse now	S.106
Supercopy	S.102
Disk-Dump	S.95
Diskettenorganisation	S.97
User-Port-Tastatur	S.92
(UPB)-Maske	S.172

VC 20

Epedemic	S.112
Video-Vorspann	S.81

Ausgabe 9/84

Bestellnummer **CB 014**

Commodore 64

Indexsequentielle Adreßdatei, S. 54 — Spring Vogel (LdM), S. 68 — Orgel/Synthesizer (AdM), S. 70 — Sprite Aid +, S. 89 — Screen Change, S. 94 — List-Stop, S. 97 — Renew, Datawandler, S. 102 — Synthetische suchen, S. 104 — Geregelter Zahlungsverkehr, S. 164

VC 20

Schiebung (GV>), S. 77 — Deuzei (8K>), S. 79 — Hardcopy 1520 (GV>), S. 87 — RS232-Interface (GV>), S. 100 — Datawandler (GV>), S. 102

Ausgabe 8/84

Bestellnummer **CB 013**

Commodore 64

Castle of Doom, S. 66 — Pac-Boy, S. 89 — Kopplung, S. 73 — User-Port-Display, S. 97 — RS232-Test, S. 77 — View BAM, S. 99 — Görlitz Hardcopy, S. 83 — Milchvieh, S. 156

VC 20

Kudiplo (3K), S. 86 — Print at Restore n (GV), S. 101

Ausgabe 7/84

Bestellnummer **CB 017**

Commodore 64

Terminalprogramm, S. 24 — Softwarekatalog, S. 72 — Russvok (SB), S. 76 — Crown No. 1, S. 80 — Space Invaders, S. 81 — 1520 Hardcopy, S. 108 — Centronics Interface, S. 110 — Kurvendiskussion, S. 116 — Copy Rel. Files, S. 132 — Autostart, S. 138 — Strubs (OP u. QP), S. 154

VC 20

Rätsel, S. 122

Bestellungen richten Sie an: M&T Buchverlag, Hans-Pinsel-Str. 2, 8013 Haar

Ausgabe 6/84

Commodore 64

Bestellnummer **CB 018**

Lehrerkalender, S. 64 — Morsetrainer, S. 72 — Supervoc, S. 69 — Grafische Darst. (SB), S. 82 — Hot Wheels, S. 92

VC 20 Bestellnummer **VC 008**

Movemaster (8K), S. 78 — Ghost Manor (GV), S.104 — Logic Disass. (3K>), S. 108 — Underground (LdM 16K), S. 120

Ausgabe 5/84

Commodore 64

Bestellnummer **CB 016**

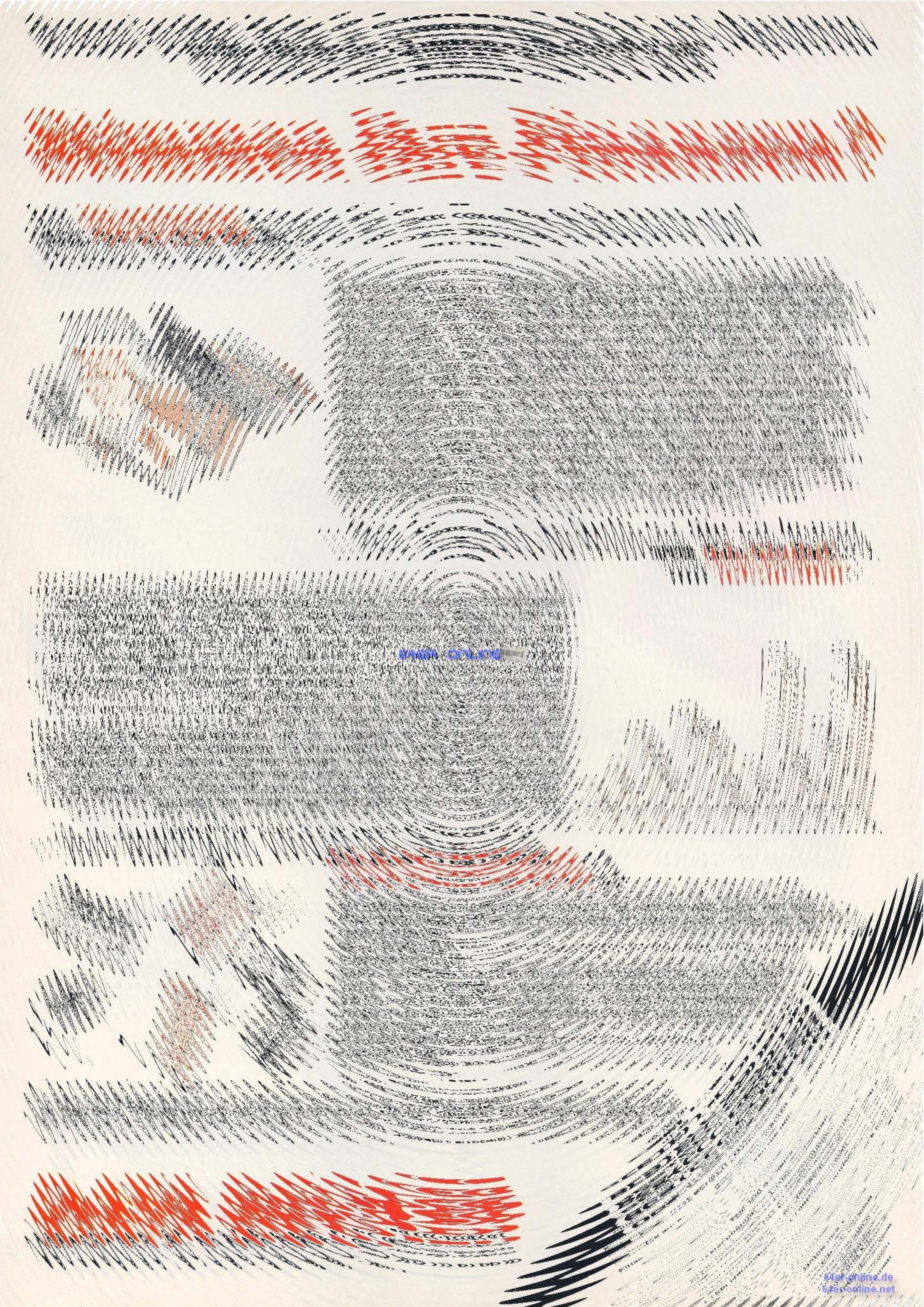
Adreß- & Telefonregister, S. 64 — Fahrsimulator, S. 82 — Schatzsucher (LdM), S. 90

VC 20 Bestellnummer **VC 007**

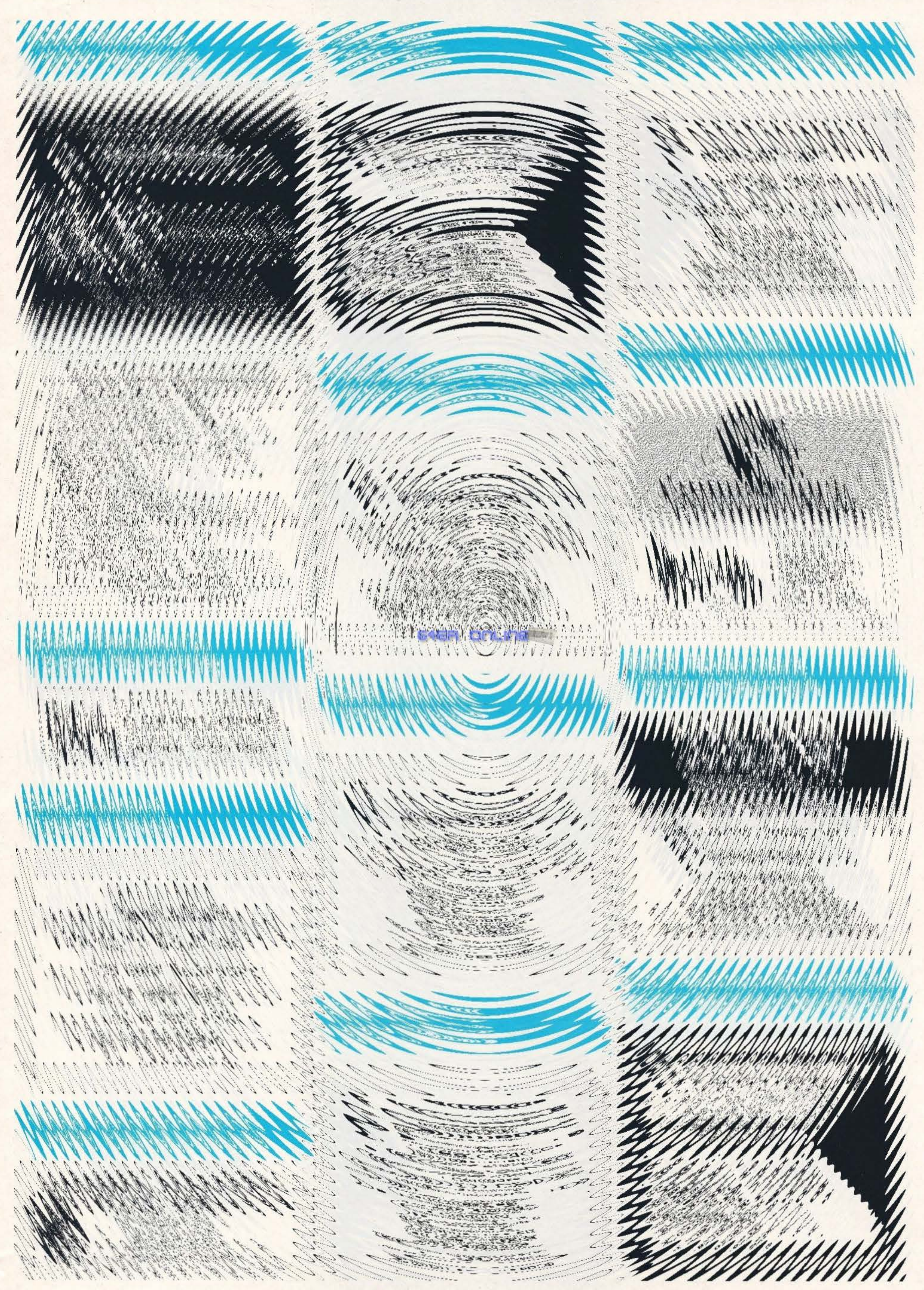
Relative Datei (8K), S. 69 — Schmatzer (GV) S. 76 — 3D-Grafik (8K), S. 78 — Rallye (28K), S. 128

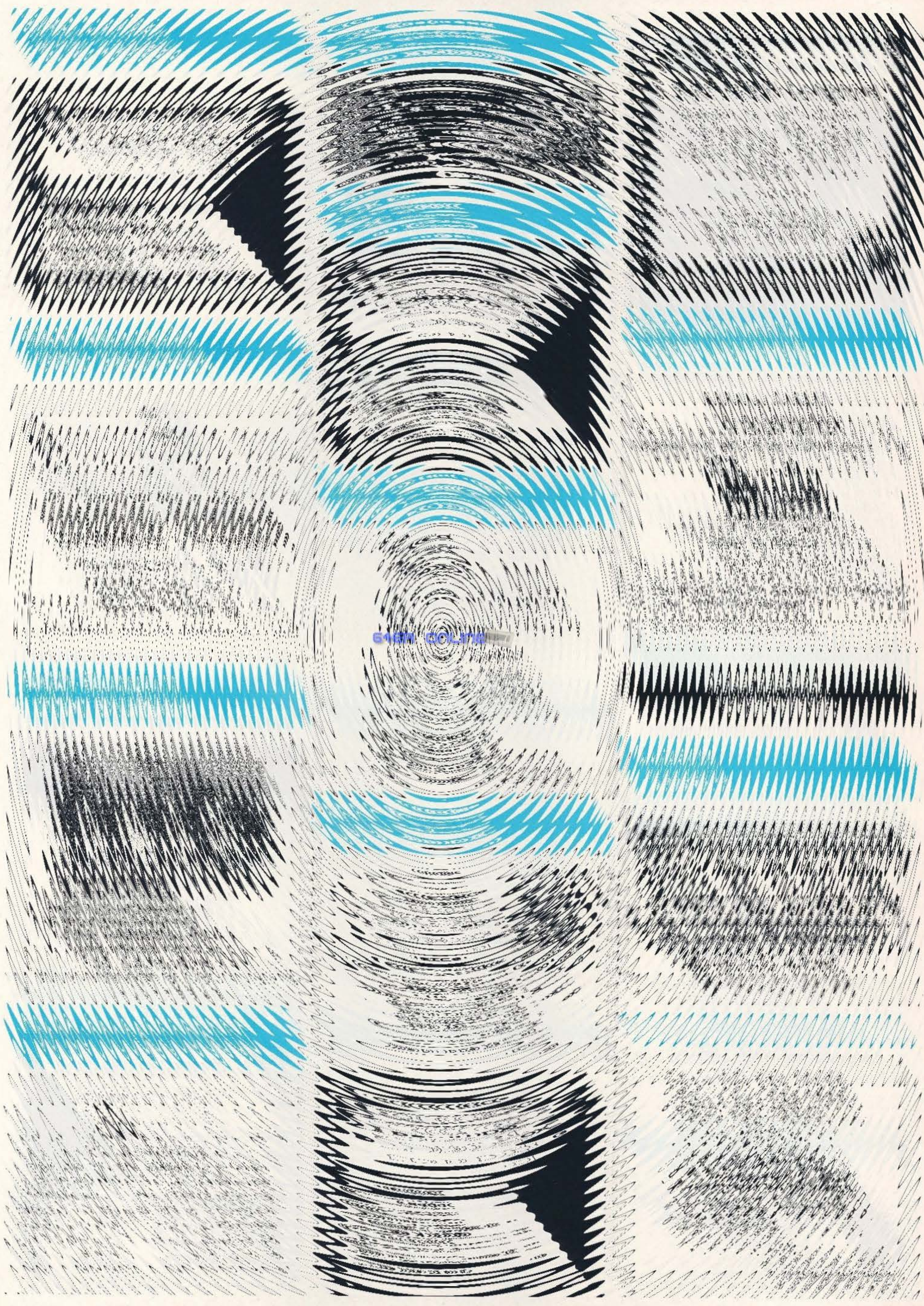
Bedeutung der Abkürzungen

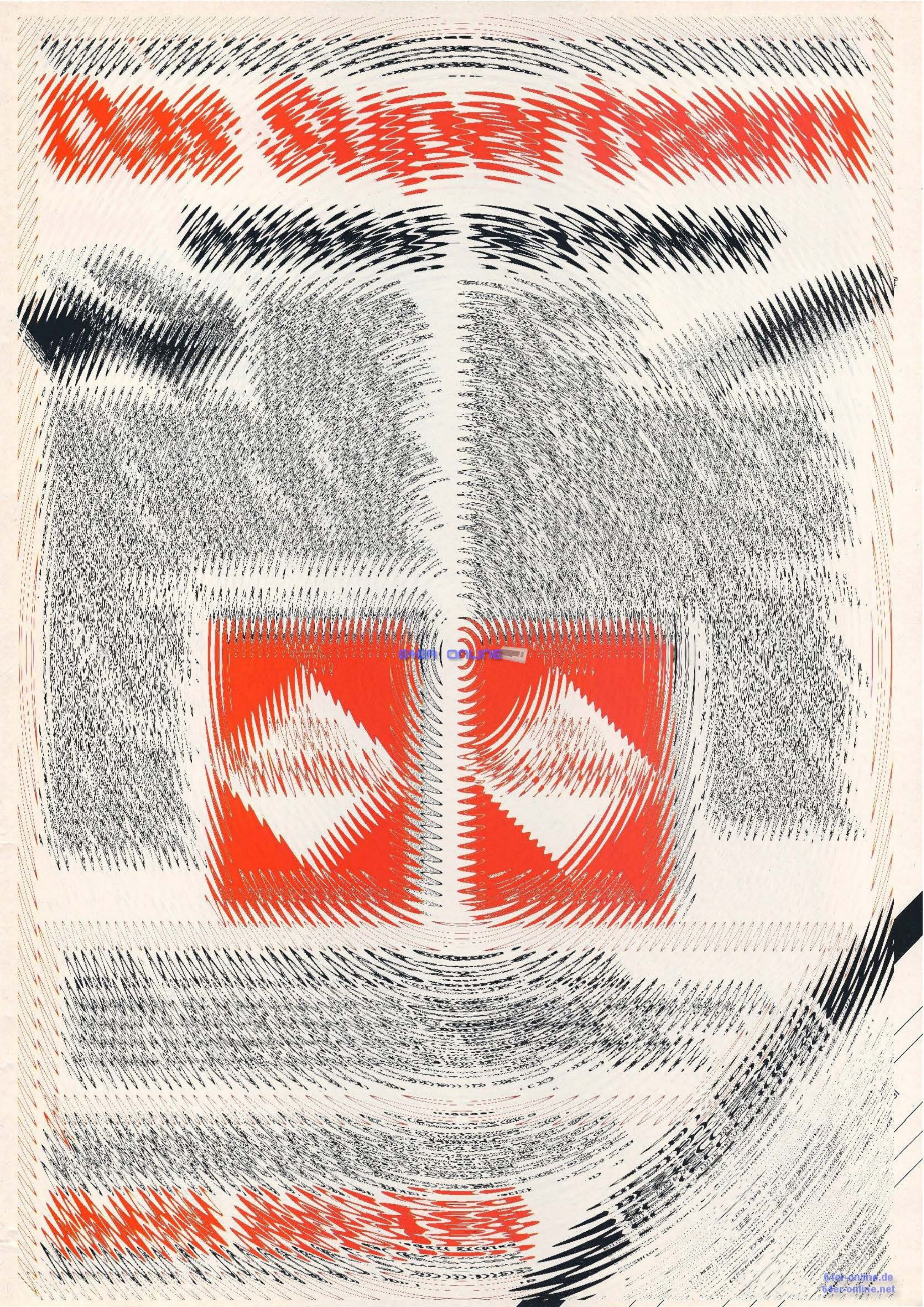
*LdM = Listing des Monats
*AdM = Anwendung des Monats
*SB = Simons Basic
*GV = Grundversion
*GV> = alle Speicherversionen können verwendet werden (einschließlich GV)
*3K = 3-KByte-Speichererweiterung wird benötigt
*8K> = Speichererweiterung größer als 8 KByte wird benötigt.
*UPB = Unterprogrammbibliothek











Unterprogrammbibliothek Exsort – Sortieren mit Komfort

Exsort zeichnet sich zum einen durch die Sortiergeschwindigkeit aus. Zum anderen werden sowohl numerische als auch alphanumerische Felder auf- oder absteigend sortiert. Ein weiterer Clou: Ein zweites Feld kann abhängig vom ersten Feld mitsortiert werden.

Jeder Programmierer ärgert sich irgendwann einmal über das langsame Basic, das vor allem beim Suchen und Sortieren stört. Gute Sortierroutinen, in Assembler geschrieben, kann nicht jeder entwickeln. Viele Sort-Programme sind aber auch sehr einseitig: Entweder sortieren sie nur aufsteigend oder lediglich alphanumerische Felder. Exsort kann beides und noch mehr.

Vorteile:

- Zirka zehnmal so schnell wie die schnellste Basic-Version.
- Die Befehle können in jedem Basic-Programm angewendet werden.
- Unterprogramme in Basic, die oft nur ein bestimmtes Feld in einer Richtung sortieren können, entfallen.
- Die Erweiterung belegt keinen Basic-Speicher.
- Beim Sortieren von Strings kommt es nicht zu einem zeitraubenden Garbage-Collect, da die Descriptoren vertauscht werden.
- Ein zweites Feld, das Informationen über das erste Feld enthält, kann mitsortiert werden.
- Das zu sortierende Feld kann numerisch oder alphanumerisch sein.

Nachteile:

— Es kann nicht mit Exbasic oder Simons Basic zusammen genutzt werden.
— Es kann nicht compiliert werden.

Weiter auf Seite 156.

```
0 REM **********************************
1 REM **MASCHINENCODE SORTIERROUTINE **
2 REM **                            **
3 REM **      VON MARCUS RICKERT    **
4 REM **********************************
100 DATA169,3,141,8,3,169,193,141,9,3,96
,160,1,177,122,1395
110 DATA201,83,240,3,76,228,167,200,177,
122,201,69,208,3,76,2054
120 DATA218,197,201,79,208,239,32,115,0,
32,115,0,32,115,0,1583
130 DATA32,253,174,32,236,196,32,36,195,
76,88,193,32,253,174,2002
140 DATA32,138,173,32,247,183,165,20,141
,0,192,165,21,141,1,1651
150 DATA192,32,253,174,32,138,173,32,247
,183,165,20,141,2,192,1976
160 DATA165,21,141,3,192,96,32,49,193,32
,253,174,32,158,183,1724
170 DATA142,4,192,224,2,144,7,162,120,16
0,197,76,250,196,32,1908
180 DATA199,195,142,6,192,140,7,192,141,
8,192,160,0,177,122,1873
190 DATA240,4,201,58,208,8,169,0,141,11,
192,76,162,193,32,1695
200 DATA253,174,32,236,196,32,36,195,32,
199,195,142,9,192,140,2063
210 DATA10,192,141,11,192,169,0,141,5,19
2,160,0,32,60,197,1502
220 DATA160,2,32,60,197,169,0,141,246,19
2,141,247,192,160,18,1957
230 DATA32,80,197,144,3,76,174,167,160,1
6,32,80,197,162,16,1536
240 DATA160,18,32,42,197,142,20,192,140,
21,192,162,20,160,246,1744
250 DATA32,154,196,240,219,48,217,173,16
,192,141,12,192,173,17,2022
260 DATA192,141,13,192,173,18,192,141,14
,192,173,19,192,141,15,1808
270 DATA192,32,239,195,160,6,174,8,192,2
32,138,162,12,32,1,1775
280 DATA197,32,47,196,240,23,72,173,4,19
2,208,6,104,16,14,1524
290 DATA76,27,194,104,48,8,162,12,32,227
,196,76,251,193,160,1766
300 DATA6,174,8,192,232,138,162,14,32,1,
197,32,47,196,240,1671
310 DATA23,72,173,4,192,208,6,104,48,14,
76,67,194,104,16,1301
320 DATA8,162,14,32,209,196,76,35,194,16
2,12,160,14,32,154,1460
330 DATA196,240,2,16,103,160,6,174,8,192
,232,138,162,12,32,1673
340 DATA1,197,134,251,132,252,160,6,174,
8,192,232,138,162,14,2053
350 DATA32,1,197,134,253,132,254,172,8,1
92,32,105,197,173,11,1893
360 DATA192,240,38,160,9,174,11,192,232,
138,162,12,32,1,197,1790
370 DATA134,251,132,252,160,9,174,11,192
,232,138,162,14,32,1,1894
380 DATA197,134,253,132,254,172,11,192,3
2,105,197,162,12,32,227,2112
390 DATA196,162,14,32,209,196,162,14,160
,12,32,154,196,48,3,1590
400 DATA76,251,193,162,14,160,16,32,42,1
97,142,20,192,140,21,1658
410 DATA192,162,18,160,12,32,42,197,142,
22,192,140,23,192,162,1688
420 DATA20,160,22,32,154,196,16,34,162,1
2,160,18,32,154,196,1368
430 DATA16,10,160,12,32,60,197,160,18,32
,60,197,173,14,192,1333
440 DATA141,18,192,173,15,192,141,19,192
,76,200,193,162,16,160,1890
450 DATA14,32,154,196,16,10,160,16,32,60
,197,160,14,32,60,1153
460 DATA197,173,12,192,141,16,192,173,13
,192,141,17,192,76,200,1927
470 DATA193,169,4,141,20,192,162,0,134,3
,134,4,32,115,0,1303
480 DATA240,63,201,58,240,59,201,36,240,
20,201,37,240,24,201,2061
490 DATA44,240,47,21,3,149,3,224,2,240,3
2,232,76,47,195,1555
500 DATA169,2,141,20,192,76,99,195,169,1
,141,20,192,165,3,1585
510 DATA9,128,133,3,165,4,9,128,133,4,76
,47,195,162,158,1354
520 DATA160,197,76,250,196,165,47,133,25
1,165,48,133,252,76,164,2313
530 DATA195,160,0,177,251,197,3,208,10,2
00,177,251,197,4,208,2238
540 DATA3,76,183,195,160,2,24,165,251,11
```

Sortierter Lebenslauf:



Ich wurde am 26.10.1967 in Köln geboren und wohne zur Zeit in Bergisch Gladbach. Ich besuche die elfte Klasse des Johann-Gottfried-Herder-Gymnasiums in Köln-Buchheim. Zum Computern kam ich durch meine Liebe zur Mathematik. Es hat mit kleinen Taschenrechnern angefangen, ging über die ersten programmierbaren Rechner (32,128,512 Schritte) zum ersten Heimcomputer (ZX81) und schließlich zum C 64, den ich im Oktober letzten Jahres als Geburtstagsgeschenk erhielt. Später folgten Floppy und Drucker 1526. Auf dem ZX81 machte ich Bekanntschaft mit Assembler (Z-80) und programmierte kleine Programme, die aber oft nur 100—300 Bytes lang waren. Erst auf dem C 64 begann ich längere Maschinenprogramme zu schreiben, da ich mit meinem Monitor arbeiten konnte.

Zur Entstehungsgeschichte des Programms:

Ich brauchte für ein Datenverwaltungsprogramm eine Sortierroutine. Ich kannte zu diesem Zeitpunkt nur das BUBBLE-Sort-Verfahren. Auf die Basic-Version programmierte ich eine in Assembler, die zwar bei Dateien unter 50 Einträgen annehmbare Zeiten lieferte, aber bei größeren Datenmengen zeitlich versagte. Dann bekam ich von einem Freund eine Kopie des QUICKSORT-Algorithmus, der schon in Basic sehr schnell ist. Nachdem ich das Sortiersystem begriffen hatte, konnte ich es in Maschinensprache umschreiben und es mit einigen Extras versehen. Das Ergebnis ist mein Programm »Exsort«, das ich inzwischen in mein Datenverwaltungsprogramm integriert habe.

Ich hoffe, daß es vielen Lesern des 64'er-Magazins eine komfortable Hilfe ist. M. Rickert

```
3,251,170,8,200,40,1841
550 DATA165,252,113,251,133,252,138,133,
251,165,251,197,49,208,212,2770
560 DATA165,252,197,50,208,206,162,143,1
60,197,76,250,196,160,4,2426
570 DATA177,251,201,1,208,1,96,162,185,1
60,197,76,250,196,24,2185
580 DATA160,6,177,251,237,2,192,8,136,17
7,251,40,237,3,192,2069
590 DATA176,7,162,0,169,18,76,59,164,24,
165,251,105,7,170,1553
600 DATA165,252,105,0,168,173,20,192,96,
24,173,16,192,109,18,1703
610 DATA192,170,173,17,192,109,19,192,74
,141,21,192,138,106,141,1877
620 DATA20,192,174,8,192,232,138,160,6,1
62,20,32,1,197,134,1668
630 DATA40,132,41,173,8,192,201,4,208,4,
138,76,162,187,174,1740
640 DATA8,192,138,168,177,40,149,97,202,
136,16,248,96,134,40,1841
650 DATA132,41,173,8,192,201,2,240,9,201
,1,240,65,138,32,1675
660 DATA91,188,96,160,1,177,40,133,3,200
,177,40,133,4,160,1603
670 DATA0,76,98,196,177,3,209,98,240,8,1
76,3,169,1,96,1550
680 DATA169,255,96,200,152,160,0,209,40,
240,8,197,97,240,4,2067
690 DATA168,76,83,196,177,40,197,97,240,
5,176,229,76,91,196,2047
700 DATA169,0,96,165,97,141,21,192,165,9
8,141,20,192,160,0,1657
710 DATA177,40,141,23,192,200,177,40,141
,22,192,162,20,160,22,1709
720 DATA189,0,192,217,0,192,208,11,189,1
,192,217,1,192,208,2009
730 DATA3,76,124,196,189,1,192,48,23,185
,1,192,48,164,56,1498
740 DATA189,0,192,249,0,192,189,1,192,24
9,1,192,144,152,76,2018
750 DATA91,196,185,1,192,16,144,76,183,1
96,56,189,0,192,233,1950
760 DATA1,157,0,192,189,1,192,233,0,157,
1,192,96,254,0,1665
770 DATA192,208,3,254,1,192,96,165,122,5
6,233,1,133,122,165,1943
780 DATA123,233,0,133,123,96,134,34,132,
35,76,71,164,133,40,1527
790 DATA189,0,192,133,113,189,1,192,133,
114,169,0,133,41,152,1751
800 DATA72,32,87,179,134,40,132,41,104,1
68,24,165,40,121,0,1339
810 DATA192,170,165,41,121,1,192,168,96,
56,185,0,192,253,0,1832
820 DATA192,72,185,1,192,253,1,192,168,1
04,170,96,174,5,192,1997
830 DATA185,0,192,157,25,192,185,1,192,1
57,136,192,232,142,5,1993
840 DATA192,96,174,5,192,202,16,2,56,96,
189,25,192,153,0,1590
850 DATA192,189,136,192,153,1,192,142,5,
192,24,96,177,251,170,2112
860 DATA177,253,145,251,138,145,253,136,
16,243,96,255,87,82,79,2356
870 DATA78,71,32,83,79,82,84,73,78,71,32
,68,73,82,69,1055
880 DATA67,84,73,79,206,65,82,82,65,89,3
2,78,79,84,32,1197
890 DATA70,79,85,78,196,87,82,79,78,71,3
2,65,82,82,65,1231
900 DATA89,32,78,65,77,197,87,82,79,78,7
1,32,73,78,68,1186
910 DATA69,216,79,78,76,89,32,79,78,69,3
2,68,73,77,69,1184
920 DATA78,83,73,79,78,32,65,82,82,65,21
7,0,255,0,255,1444
930 DATA0,255,32,255,0,32,115,0,32,115,0
,32,115,0,32,1015
940 DATA253,174,32,236,196,32,36,195,32,
49,193,32,199,195,142,1996
950 DATA6,192,140,7,192,141,8,192,32,253
,174,32,158,173,162,1862
960 DATA2,160,0,32,154,196,16,7,162,174,
160,197,76,250,196,1782
970 DATA173,8,192,201,2,240,20,32,141,17
3,173,8,192,201,4,1760
980 DATA240,23,32,170,177,133,97,132,98,
76,57,198,32,163,182,1810
990 DATA133,97,165,34,133,98,165,35,133,
99,174,8,192,232,138,1836
1000 DATA160,6,162,0,32,1,197,142,9,192,
140,10,192,174,9,1426
```

Listing 1. »Exsort data«

Unterprogrammbibliothek:

```
1010 DATA192,172,10,192,32,47,196,240,42
,162,0,160,2,32,154,1633
1020 DATA196,240,26,162,0,32,227,196,56,
173,9,192,109,8,192,1818
1030 DATA141,9,192,173,10,192,105,0,141,
10,192,76,75,198,169,1683
1040 DATA255,160,255,76,134,198,173,1,19
2,172,0,192,32,145,179,2164
1050 DATA169,0,133,13,133,14,169,73,133,
69,169,78,133,70,32,1388
1060 DATA231,176,166,71,164,72,32,212,18
7,76,174,167,255,0,255,2238
1100 ZN=100:Z=49400:RESTORE:REM ** EINLE
SEN DER WERTE **
1110 PR=0
1120 FORS=0TO14:READX:PR=PR+X:POKEZ+S,X:
NEXTS
1130 READX:IFX=PRTHEN1150
1140 PRINTCHR$(14)"■RUEFSUMMEN--EHLER I
N ◆EILE";ZN:END
1150 ZN=ZN+10:Z=Z+15
1160 IFZ<>50855THEN1110
1170 PRINT"FERTIG"
1180 INPUT"■CASSETTE(1) ODER DISKETTE(8)
";PN
1190 POKE251,PN:REM ** ABSPEICHERN ALS A
BSOLUTPROGRAMM **
1200 POKE252,PEEK(45):POKE253,PEEK(46)
1210 POKE43,248:POKE44,192
1220 POKE45,172:POKE46,198
1230 IFPEEK(251)=1THENSAVE"EXSORT",1,2
1240 IFPEEK(251)<>1THENSAVE"@:EXSORT",PE
EK(251)
1250 POKE43,1:POKE44,8:POKE45,PEEK(252):
POKE46,PEEK(253)
READY.
```

Listing 1. »Exsort data« (Schluß)

1. Befehl »so«
Syntax: so, (feldname), (anfangsindex), (endindex), (sortierungsrichtung)

Dieser Befehl sortiert ein beliebiges eindimensionales Feld innerhalb von zwei Grenzen mit einer vom Benutzer gewählten Sortierungsrichtung.

Beispiel 1:
Das Feld heißt ax$, alphanumerisch aufsteigend sortieren (von Index 100 bis Index 5000).
Befehl: so,ax$,100,5000,1 (1 = aufsteigend)

Beispiel 2:
Das Feld heißt qe%, numerisch absteigend sortieren (von Index 0 bis zu dem Index, der in der Variable »en« enthalten ist).
Befehl: so,qe%,0,en,0 (0 = absteigend)

Option: Manchmal ist es notwendig, daß Daten, die in einem zweiten Feld vorhanden sind, entsprechend dem ersten Feld sortiert werden.
Syntax: so,(feldname 1), (anfangsindex), (endindex), (sortierungsrichtung), (feldname2)
Beispiel: Das Feld fe$ soll alphanumerisch aufsteigend von Index 0 bis Index 10 sortiert werden. Die Daten in dem Realfeld »nr« sollen entsprechend dem ersten Feld sortiert werden.
Befehl: so,fe$,0,10,1,nr

2. Befehl: »se«
Syntax: se,(feldname),(anfangsindex),(endindex),(element)

Dieser Befehl durchsucht ein beliebiges eindimensionales Feld innerhalb von zwei Grenzen nach einem Element.
Beispiel: Es soll die Zahl - 12 in dem Feld rt% von Index 0 bis Index 100 gesucht werden.
Befehl: se,rt%,0,100, - 12

Wenn das Element gefunden wird, enthält die Variable »in« den jeweiligen Index. Wird das Element nicht gefunden, so enthält »in« den Wert —1.

```
0 IFK=0THENK=1:LOAD"EXSORT?",8,1:REM LAD
EN VON EXSORT
1 SYS49400:REM STARTEN VON EXSORT
100 REM ***********************
110 REM ***   EXSORT DEMO   ***
120 REM ***********************
130 REM
140 REM *****************
150 REM * 1.BEFEHL: "SO" *
160 REM *****************
165 PRINT"■ERSTER BEFEHL: 'SO'"
170 INPUT"■ZAHL DER ZU SORTIERENDEN ELEM
ENTE";A
175 PRINT"■ANFUELLEN DES FELDES 'ZA' MIT
 ZUFALLS-     ZAHLEN"
180 DIMZA(A)
190 REM *** DAS FELD ZA WIRD MIT ZUFALLS
ZAHLEN BELEGT ***
200 FORS=1TOA
210 :ZA(S)=RND(1)*10000-5000
220 NEXTS
230 IT=TI:REM ZEIT SPEICHERN
240 PRINT"■SORTIERBEGINN"
250 REM
260 REM *** AUFRUF DES BEFEHLS "SO" ***
270 SO,ZA,1,A,1
280 REM   SO   =   BEFEHL
290 REM   ZA   =   FELDNAME
300 REM   1    =   ANFANGSINDEX
310 REM   A    =   ENDINDEX
320 REM   1    =   SORTIERUNGSRICHTUNG(A
UFSTEIGEND)
330 REM
340 IT=TI-IT
350 PRINT"■SORTIERENDE"
355 FORS=1TO1000:NEXTS
360 REM *** AUSGABE DER SORTIERTEN ELEME
NTE ***
370 FORS=1TOA
380 :PRINTS,TAB(6)ZA(S)
390 NEXTS
400 PRINT"■◆EIT:"IT/60"SEC"
410 PRINT"■BITTE TASTE DRUECKEN"
420 GETT$:IFT$=""THEN420
430 REM
440 REM *****************
450 REM * 2.BEFEHL: "SE" *
460 REM *****************
470 REM
480 CLR:DIMFE$(10000)
485 PRINT"■ZWEITER BEFEHL 'SE'"
490 REM ** IN 50 BELIEBIGE ELEMENTE   **
500 REM ** DES FELDES FE$ WIRD DAS    **
510 REM ** WORT "HALLO" GESCHRIEBEN   **
520 REM
525 PRINT"■IN 50 BELIEBIGE ELEMENTE VON
FE$ WIRD   'HALLO' GESCHRIEBEN"
530 FORS=1TO50
540 :FE$(RND(1)*10000)="HALLO"
550 NEXTS
560 PRINT"■IN FOLGENDEM ELEMENTEN STEHT
'HALLO':"
570 REM
580 REM ** AUSDRUCKEN JEDES INDEXES **
590 REM ** IN DEM "HALLO" STEHT     **
600 REM
610 IN=-1:IT=TI
```

Listing 2. »Exsort demo«

Exsort — Sortieren mit Komfort

```
620 REM ** AUFRUF DES BEFEHLS "SE" **
630 SE,FE$,IN+1,10000,"HALLO"
640 REM  SE       =   BEFEHL
650 REM  FE$      =   FELDNAME
660 REM  IN+1     =   ANFANGSINDEX
670 REM  10000    =   ENDINDEX
680 REM  "HALLO" =   ELEMENT
690 REM ** BEI RUECKKEHR AUS "SE"  **
700 REM ** ENTHAELT "IN" DEN INDEX **
710 REM ** ODER (WENN DAS ELEMENT  **
720 REM ** NICHT GEFUNDEN WURDE)   **
730 REM ** DEN WERT -1             **
740 IFIN=-1ORIN=10000THEN760
750 PRINTIN,:GOTO630
760 PRINT:PRINT"▒ZEIT:"(TI-IT)/60"SEC"
770 PRINT"▒BITTE TASTE DRUECKEN"
780 GETT$:IFT$=""THEN780
790 REM
800 REM ****************************
810 REM * 1.BEFEHL 'SO' MIT OPTION *
820 REM ****************************
830 REM
835 PRINT"▒ERSTER BEFEHL MIT OPTION"
840 DATANULL,ZWEI,VIER,SECHS,ACHT,ZEHN,E
INS,DREI,FUENF,SIEBEN,NEUN
850 DATA0,2,4,6,8,10,1,3,5,7,9
860 CLR:DIMNR(10),NR$(10)
870 REM ** EINLESEN IN FELD NR$ **
880 FORS=0TO10
890 :READX$:NR$(S)=X$
900 NEXTS
910 REM ** EINLESEN IN FELD NR  **
920 FORS=0TO10
930 :READX:NR(S)=X
940 NEXTS
950 REM ** AUSGABE FELD VOR SORTIERUNG *
*
960 PRINT"▒INDEX NR$ VORHER NR * NR$ NAC
HHER NR"
970 FORS=0TO10
980 :PRINTS;TAB(6)NR$(S)TAB(16)NR(S)
990 NEXTS
1000 REM ** AUFRUF DES BEFEHL "SO" MIT O
PTION **
1010 SO,NR$,0,10,0,NR
1020 REM SO   =   BEFEHL
1030 REM NR$  =   FELDNAME 1
1040 REM 0    =   ANFANGSINDEX
1050 REM 10   =   ENDINDEX
1060 REM 0    =   SORTIERUNGSRICHUTNG(AB
STEIGEND)
1070 REM NR   =   FELDNAME 2
1080 REM
1090 PRINT"▒SORTIEREN VON NR$ ABSTEIGEND
"
1095 PRINT"▒NR WIRD ENTSPRECHEND MITSORT
IERT"
1097 PRINT"▒BITTE |ASTE DRUECKEN"
1098 GETT$:IFT$=""THEN1098
1100 REM ** AUSGABE FELD NACH SORTIERUNG
 **
1110 PRINT"▒▒▒▒";
1120 FORS=0TO10
1130 :PRINTTAB(22)NR$(S)TAB(33)NR(S)
1140 NEXTS
1150 PRINT"▒▒▒▒▒▒▒"
READY.
```

Listing 2. »Exsort demo« (Schluß)

Fehlermeldungen:

- type mismatch:
  Sie versuchten, einen String in einem numerischen Feld zu suchen (oder umgekehrt).
- wrong index:
  Beim Suchen war der Anfangsindex größer als der Endindex.
- bad subscript:
  Index außerhalb des zulässigen Bereiches.
- only one dimension array:
  Sie können nur eindimensionale Felder durchsuchen oder sortieren.
- array not found:
  Das Feld war nicht durch einen DIM-Befehl dimensioniert worden.
- wrong array name:
  Geben Sie bitte nur die ersten beiden Buchstaben des Feldnamen ein (plus % oder $ wenn nötig). Es wird dann sicher funktionieren.
- wrong sorting direction error:
  Sie haben einen anderen Wert als 0 oder 1 als Sortierungsrichtung angegeben.

Zu den Programmen:

Listing 1

Das Programm »Exsort data« erstellt das Maschinenprogramm aus DATA-Zeilen und speichert es als »Exsort«-Absolutprogramm auf Diskette oder Kassette. Sie können es dann jederzeit durch LOAD »Exsort«, 8,1 absolut laden. Dabei geht ein Basic-Programm nicht verloren (siehe auch Listing 2, Demo-Programm).

Listing 2

Das Programm »Exsort demo« lädt das Absolutprogramm »Exsort« nach und startet es. Danach folgt eine Demonstration der beiden Befehle.

Um »Exsort« zu laden, muß Zeile 0 des Basic-Programms lauten:

0 if k=0 then k=1 : load"exsort",8,1 (für Diskette)

0 if k=0 then k=1 : load"exsort",1,1 (für Kassette)

In Zeile 1 muß stehen:

1 sys 49400

Da die Erweiterung nur einmal geladen und gestartet werden muß, kann sie bei späteren Starts des Programms übersprungen werden. (Marcus Rickert/gk)

Fortsetzung von Seite 31.

Wie super ist die Supergraphik?

gramme angehängt werden; so ist zum Beispiel ein komfortables Nachladen der Spritedaten möglich. RENUM ist ein Zeilennumerierungs-Befehl; er gleicht auch alle Sprungadressen hinter GOTO, GOSUB an. Ihre Funktionstasten können Sie mit KEY belegen. Sie sind relativ sinnvoll vorbelegt; so ist mit F5/F7 eine Umschaltung zwischen Textseite und Grafikseite möglich. DTASET ist ein gezieltes RESTORE; da hier auch arithmetische Ausdrücke vorkommen dürfen, ist eine Angleichung beim RENUM-Befehl nicht implementiert. Neben der Joystickabfrage ist auch eine Paddleabfrage über PADDLE realisiert worden. Als letztes noch der Befehl POS=; mit ihm kann im Textmodus der Cursor einfach auf beliebige Spalten und Zeilen gesetzt werden.

Ein anderes Detail, das mir allerdings sehr gut gefallen hat: Bei etwaigen Fehlermeldungen wird von der Grafikseite wieder automatisch auf die Textseite umgeschaltet. Das hält Sie beim Austesten eines Programms immer auf dem Laufenden.

Supergraphik 64 ist ein Programm von Data Becker für 99 Mark, das einerseits durch seine Befehlsvielfalt, andererseits durch einige kleine Details beeindruckt. Einige Mängel schwächen den positiven Eindruck allerdings ein wenig ab. Man sollte aber auf jeden Fall beim Kauf einer Grafikerweiterung Supergraphik 64 in Erwägung ziehen, insbesondere, wenn man es nicht so sehr auf komfortables Sprite-Handling, sondern eher auf ein gutes Werkzeug für hochauflösende Grafiken abgesehen hat.

(Boris Schneider/aa)



EINZEILER-WETTBEWERB:

Wir wußten gar nicht, was auf uns zukommen sollte, als die Idee Täglich gingen zirka 20 bis 40 Einzeiler bei uns ein.

Man glaubt gar nicht, wie interessant manche Einzeiler sind. Hier werden Probleme in eine Zeile gesteckt, bei denen manche Programmierer etliche KByte benötigen. Ein Beispiel ist das Mini-Orgel-Programm. Vielleicht finden Sie, daß hier mit einem Trick gearbeitet wurde, aber Einzeiler ist Einzeiler. Sehr viele Programme lassen sich nur eingeben, wenn die Abkürzungen der Basic-Befehle benutzt werden. Sie stehen im Anhang D Ihres C 64 Handbuches. Doch schauen Sie sich die folgenden Mini-Programme an, von denen man sagen kann: Klein — aber oho! Jedes von ihnen wird deshalb mit 100 Mark honoriert.

Merge (C 64)

```
10 a=peek(45)+256*peek(46)-2:poke44,a/25
6:poke43,a-peek(44)*256:print"prg laden
& pO43,1:pO44,8
```

Möchte man zwei Programme zu einem einzigen zusammenfassen, gab es oft nur eine Möglichkeit: das kürzere an das Ende des längeren zu tippen. Manche längere Hilfsprogramme vollbringen durchaus ein richtiges MERGE. Es geht aber auch ganz kurz. (Andreas Gast)

Directory laden, ohne Basic-Programm zu zerstören (C 64/VC 20)

```
0 get#1,a$:a=asc(a$+"▒"):printchr$((a=13
0and13or((31<aanda<95)anda));:goto0
```

Zunächst eine Bemerkung zu einem dem Programmierer sehr bekannten Problem. Hat man nun, mit Ach und Krach, ein wichtiges Programm in Basic geschrieben, und man will seine Arbeit mit dem Abspeichern dieses Programmes beenden, so kann es vorkommen, daß man seine freien Disketten nicht wiedererkennt. Also muß man sich die Disketten listen. Mit LOAD"$",8:LIST würde aber das eigene Basic-Programm gelöscht und die ganze Arbeit wäre umsonst. Wenn man dagegen diese kleine Zeile eingibt, und zwar so, daß sie vor dem eigenen Programm liegt, läßt sich das Directory listen, ohne das im Speicher befindliche Programm zu zerstören.
Zum Starten:
OPEN 1,8,2,"$":GOTO 0

Beschreibung

OPEN 1,8,2,"$"	eröffnet eine sequentielle Datei (hier das Directory) zum Lesen.
GET#1,A$	holt ein Byte vom Disketten-Puffer (Buffer).
A=ASC(A$+ "SHIFT/HOME")	wandelt ASCII-Zeichen nach Zahl (von 0-255). Wenn A$ eine Länge von 0 hat, gibt es keine Fehlermeldung, da der Ausdruck trotzdem eine Länge von 1 behält.
PRINT CHR$((A=130 AND 13 OR ((31<A AND A<95) AND A)));	druckt für A das zugehörige Zeichen, wenn sich A zwischen 32 und 95 bewegt. Wenn A=130 ist, wird Return ausgegeben. Durch die Formel innerhalb der Characterstring-Klammer werden alle Steuerzeichen und Grafiksymbole »herausgefiltert«. Es bleiben dann nur die Zahlen, Buchstaben und Satzzeichen, die ausgedruckt werden.

Weitere Anwendungen:
Da man alle auf der Diskette befindlichen Programme als sequentielle Datei lesen kann, ist es ohne weiteres möglich, mit dem obigen Programm alle Kommentare, Inhalte von Printanweisungen und Texte des auf der Diskette befindlichen Programms auf den Bildschirm zu bringen. Zum Starten eröffnet man dann die Datei nach dem folgenden Schema:
OPEN 1,8,2,"filename"
und startet das Programm mit GOTO 0.

Wenn die Ausgabe beendet ist, wird mit der RUN/STOP-Taste die Endlosschleife verlassen.

(Reinhard Abdel-Hamid)

Speicherblockverschiebung — Blitzschnell (C 64)

```
1 poke95,al:poke96,ah:poke90,el:poke91,e
h:poke88,nl:poke89,nh:sys41919
11 rem beispiel:
12 poke95,0:poke96,160:poke90,0:poke91,1
92:poke88,0:poke89,192:sys41919
13 rem ↑↑ basic ins ram ↑↑
```

Dieses Programm dient zur Übertragung von Speicherblöcken. Die Variablen mit L sind also jeweils das Low-Byte, die mit H das High-Byte der Adresse.
Sie lassen sich für eine Adresse X so berechnen:
AL=X-256*INT(X/256): AH=INT(X/256)

Dieser Einzeiler benützt die Blockverschieberoutine des ROMs. Er ist zum Beispiel nützlich, um das Basic beziehungsweise das Betriebssystem vom ROM ins RAM zu verlegen oder um den Zeichengenerator zu kopieren (!).

(Jens Baas)

DI-AS, ein grafischer Disassembler für C 64

```
5 print"▒▒1▒b▒q▒(▒HP▄▒ :"peek(3)"▒ ":pok
e41,5:geta$:poke3,peek(3)-(a$="▒")+(a$="
▒"):sys1024:goto5
```

Der Speicherbereich des C 64 läßt sich in 256 Seiten (Pages) à 256 Byte aufteilen. DI-AS interpretiert die Speicherinhalte als Bildschirmcode und stellt die 64 KByte des C 64 Seite für Seite auf den Zeilen 7-13 des Bildschirms dar. Die Seitennummer wird am oberen Bildschirmrand angezeigt.
Die Bedienung:

Bevor das Programm geladen und gestartet wird, muß mit SYS64738 unbedingt ein Reset durchgeführt werden, da sonst möglicherweise die Zeiger der Zeropage nicht korrekt initialisiert werden. Es empfiehlt sich, den Bildschirm mit CLR zu löschen, um die Übersichtlichkeit zu erhöhen.
Das Programm wird mit RUN gestartet.
Dem Benutzer stehen nun folgende Optionen offen:
1. Mit CURSOR-RIGHT im Speicher vorwärts blättern
2. Mit CURSOR-DOWN im Speicher rückwärts blättern
3. Mit R/S POKE3, NR :CLR: RUN die Seite NR betrachten
Fehlerbehandlung:

Es stehen umfangreiche Fehlerbehandlungen zur Verfügung: Der Versuch, Seiten kleiner als 0 oder größer als 256 anzusehen wird mit »illegal quantity error in 5« quittiert. Mit CLR: RUN wird das Programm normal wiedergestartet.

Beachten Sie sorgfältig die Hinweise zum korrekten Eintippen des Programms!

DIE TOP 10

zum Einzeiler-Wettbewerb geboren war. Und diese Flut hat noch kein Ende.

Gehen Sie die Anwendungsbeispiele durch, um sich mit der Speicheraufteilung Ihres C 64 vertraut zu machen. Da das Programm die 80 Zeichen des Basic-Editors benötigt, muß der Cursor erst eine Zeile hochgefahren werden, bevor diese mit RETURN in den Programmspeicher übernommen wird.

Steuerzeichen:
Reverses s = Home
Reverses r = RVSON
Reverses R = RVSOFF
hinter dem P: Commodore O
die Restlichen: Cursor links, rechts, unten

Anwendungsbeispiele:

Wenn Sie das Programm zum ersten Mal starten, so wird Ihnen wahrscheinlich Seite 170 angezeigt. Drücken Sie solange auf CURSOR-DOWN, bis Sie bei Seite 0 angelangt sind. Falls Sie zu weit gefahren sein sollten, und das Programm den Fehler meldet, so starten Sie es einfach noch einmal. Sie sehen jetzt die Zeropage als Bildschirmcodes. Nun gehen Sie die Seiten Schritt für Schritt durch. Dabei können Sie beobachten:

Seite 0	: In Zeile 5 die Real-time Jiffy clock, die vom Betriebssystem versorgt wird.
Seite 1	: Unterste Zeilen = der Mikroprozessor System-Stack
Seite 2	: Oberste 2 Zeilen = Input Buffer. Hier stehen Ihre letzten Eingaben.
Seite 4-7	: Der Bildschirmspeicher selbst bildet sich ab. Halten Sie das Programm an, schreiben ein paar Sätze in die untere Bildschirmhälfte und schauen, was sich verändert hat. Außerdem zu sehen sind die Sprite-Pointer.
Seite 8	: Hier steht unser Basic-Programm. Erkennen Sie es wieder?
Seite 160	: Das CBM-BASIC-ROM.*
Seite 161	: Die Liste der Basic-Befehle*
Seite 162	: Die Fehlermeldungen*
192-207	: Leerer RAM für Ihre Maschinenprogramme
Seite 208	: Der VIC. Was da so flackert ist das VIC-Control-Reg. D011 und das Read-Raster-Reg. D012.
212-215	: Der SID
216-219	: Ein wüstes Bild. Das soll der Farb-RAM sein? Nur die Low-Nibbles behalten ihre Werte!!
220-223	: Die CIA. Beobachten Sie die Timer.
224-227	: Basic-Befehle: open, close, sin, cos, tan und andere
228-256	: Das Betriebssystem.*

*Anmerkung: Manches läßt sich erst entdecken, wenn durch gleichzeitiges Drücken von SHIFT und COMMODORE der Zeichensatz umgeändert wird. Nun nehmen Sie doch einmal Simons Basic oder Spiele und schauen, wo der Text, die Befehle etc. stehen! (Andreas Carl)

Scrollen mit bleibendem Text (C 64)

```
1 fori=40960to49151:pokei,peek(i):next:f
ori=57344to65535:pokei,peek(i):next:poke
59639,10:poke1,53
```

Dieses kleine Programm bewirkt, daß nicht mehr der gesamte Bildschirm gescrollt wird. Wieviel Zeilen am oberen Rand fest stehen bleiben, bestimmt die Zahl, die Sie in 59639 POKEn. So kann man eine Information, die für ein Programm wichtig ist, auf dem Bildschirm festhalten. Oder die Kopfzeilen einer Tabelle, etc.

Mein Beispiel mit 0 gePOKEt hält eine Zeile fest. Eine 1 würde 2 Zeilen festhalten, etc. Abgestellt wird das Programm einfach durch »POKE 1,55«. Es muß noch darauf hingewiesen werden, daß dieses Programm, obwohl es so kurz ist, sehr lange bis zur ersten Ausführung braucht. (Peter Eckart)

Maschinenprogramme abspeichern (C 64)

```
1 sys(57812)a$,x:poke193,ls:poke194,hs:p
oke174,le:poke175,he:sys62957
```

Mit folgendem Einzeiler können Maschinen-Programme sehr einfach gespeichert werden: (ohne Basic-Pointer zu verstellen)

Für die Berechnung von LE und HE muß die Endadresse + 1 genommen werden.

Als erstes Beispiel soll ein Programm von 20000 — 22000 unter dem Namen »BEISPIEL 1« auf Diskette gespeichert werden.

SYS(57812)"BEISPIEL 1", 8:POKE193,32:POKE194,78:POKE 174,241:POKE175,85:SYS62957

Als zweites Beispiel soll ein Programm von $C000 — $C37E unter dem Namen "BEISPIEL 2" auf Kassette gespeichert werden.

SYS(57812)"BEISPIEL 2",1:POKE193,0:POKE194,192:POKE174, 127:POKE175,195:SYS62957

Falls der Einzeiler infolge eines längeren Programm-Namens zu lang werden sollte, können die POKE-Befehle selbstverständlich durch P, Shift O abgekürzt werden. (Markus Eicher)

Das kürzeste Heimcomputerorgelprogramm der Welt (C 64)

```
1 print"▒▒)om▒x▒T) ┌m▒e▒T┐▒ ▒P ▒p▒1▒d▒TN▒
 ▒P%K ┌▒s)?▒m▒a▒TH1▒d▒T-▒r▒PM▒r▒PPV ┌▒▒":s
ys4↑5
```

Um in Basic ein Maschinensprachprogramm einzulesen, schreibt man normalerweise einen Basic-Lader, der in einer FOR/NEXT-Schleife mehrere DATAs einliest und in einen freien Speicherbereich POKEt, um dann das Programm mit SYS zu starten. Da diese Befehle und die DATAs sehr viel Platz benötigen, können so in einer Zeile kaum umfangreiche Programme geschrieben werden. Deshalb haben wir eine Möglichkeit entwickelt, den Basic-Lader zu umgehen: Wir PRINTen das Maschinenspracheprogramm in die erste Bildschirmzeile und starten das Programm mit SYS 1024. Mit diesem Trick ist es uns gelungen, ein Programm von 45 Maschinensprachebytes in einer Basic-Zeile zu laden und zu starten.

Das Programm »MICROSOUND« ist das, unseres Wissens, kürzeste Heimcomputerorgelprogramm der Welt. Es ist 86 Basic-Bytes lang und erzeugt ein Maschinenspracheprogramm von 45 Byte. Davon werden 14 Byte für Grafik-Effekte und 3 Byte für eine Programmende-Abfrage genutzt. Da der Einzeiler bereits beim geringsten Tippfehler nicht mehr richtig funktioniert, haben wir zum Vergleich den normalen Basic-Lader und das Monitor-Listing mit Erklärungen beigefügt.

Das Musik-Programm funktioniert im Wesentlichen dadurch, daß die Zahl der gedrückten Taste (PEEK (203)) mit 63 AND-verknüpft in das Hi-Byte der 1. Stimme des SID 6581 gebracht wird. Der Grafik-Effekt kommt durch einen Vergleich mit dem Rasterregister in 53266 zustande.

MICROSOUND ist für den Commodore C 64 entwickelt. Das Programm wird mit RUN gestartet und mit INST/DEL beendet. Es können ohne weiteres Basic-Zeilen angefügt werden, die dann nach INST/DEL abgearbeitet werden. Wenn das Maschinenspracheprogramm in der erste Zeile nicht mehr sichtbar sein soll, muß vorher die Zeichenfarbe

EINZEILER-WETTBEWERB: DIE TOP 10

auf die Bildschirmfarbe umgestellt werden. Der Basic-Lader kann auch ohne weiteres auf Bereiche außerhalb des Bildschirms gesetzt werden, indem man die Variable »anfang« entsprechend ändert. Alle Tasten, außer CTRL, COMM., SHIFT und RESTORE sind mit Noten belegt, die überwiegend so angeordnet sind, daß die linken Tasten tiefere Töne spielen als die rechten. Die Funktions- und Cursor-Tasten sind Bässe.

Tips zur Eingabe dieses Programms:

1. Schalten Sie vor der Eingabe durch SHIFT/COMM. auf Kleinschrift.
2. Geben Sie ? statt print und sY4(Pfeil n. oben)5 statt sys1024 ein und verzichten Sie auf unnötige Leerzeichen.
3. SPACEs bitte ohne COMM. oder SHIFT-Tasten zu drücken eingeben. Ausnahme: Das letzte SPACE **muß** ein SHIFT/SPACE sein.
4. Achten Sie genau auf Groß- und Kleinschreibung! Das Programm kann beim kleinsten Tippfehler abstürzen!

MICROSOUND als Basic-Lader:

```
10 rem***microsound***
20 anfang=1024
30 for x=0 to 44:read a:poke an+x,a:next
40 sys an
100 data 169, 143, 141, 24, 212, 169, 240, 141, 5, 212, 238
110 data 32, 208, 160, 16, 140, 4, 212, 206, 32, 208, 165
120 data 203, 240, 19, 41, 63, 141, 1, 212, 200, 140, 4
130 data 212, 173, 18, 208, 205, 18, 208, 208, 214, 240, 246, 96
```

Der Basic-Lader hat gegenüber dem Einzeiler den Vorteil, daß man ihn leichter abtippen und ohne weiteres in einen anderen freien RAM-Bereich verlegen kann. Wenn Sie richtig abgetippt haben, müssen beide Programme identisch arbeiten.

Monitor Listing:

```
0400 lda  #$8f;
0402 sta  $d418;    Lautstärke (si+24)
0405 lda  #$f0;
0407 sta  $d405;    Halten d. Tones (si+5)
040a inc  $d020;    Rahmenfarbe um 1 erhöhen
040d ldy  #$10;
040f sty  $d404;    Wellenform (si+4) = 16
0412 dec  $d020;    Rahmenfarbe um 1 vermindern
0415 lda  $cb;      gedrückte Taste (peek(203))
0417 beq  $042c;    wenn $cb=null (inst/del gedrückt)
                    dann Ende
0419 and  #$3f;     wenn $cb=64 dann akku=0
041b sta  $d401;    akku nach hi-byte von Stimme 1 (si+1)
041e iny;
041f sty  $d404;    Wellenform (si+4) = 17
0422 lda  $d012;    Rasterregister (v+18)
0425 cmp  $d012;    (Grafik-Effekt durch registergesteuer-
                    te Verzögerung)
0428 bne  $0400;
042a beq  $0422;
042c rts;           Basic-Rücksprung
```

(Klaus und Stefan Holthausen)

Elektronisches Klavier für den VC 20

```
1 poke36878,15:geta$:a=val(a$):poke36876
,127+(10-(a>1)+(a>3)+(a>5)*.4+(a>6)*.3+(
a>7)*.6)*a:goto1
```

Das Programm läuft nur auf dem VC 20 und verwandelt den Computer in ein elektronisches Klavier. Die Tasten 1 bis 8 dienen dabei als Klaviatur, auf der man in etwa eine D-Dur Tonleiter spielen kann.

Der Computer gibt beim Niederdrücken der Tasten 1 bis 9 jeweils einen kurzen Ton aus. Die Zahlen 1 bis 8 entsprechen dabei den Tönen einer D-Dur Tonleiter, also d, e, fis, g, a, h, cis, d. Das Betätigen der Taste 9 ergibt zwar auch einen Ton, der sich aber nicht mehr in die Tonleiter einreiht. Andere Tasten haben, außer der RUN/STOP-Taste, keine Funktion.

Basic-Programm:

1 POKE36878,15:GETA $:A=VAL(A$):POKE36876,127+(10—(A>1)+(A>3)+(A>5)*.4+(A>6)*.3+(A>7)*.6)*A:GOTO1

Beschreibung des Programms:

POKE36878,15: Die Lautstärke des VIC-Chips wird gesetzt.

GETA$:A=VAL(A$): Der Variablen A wird ein Wert zwischen 0 bis 9 zugewiesen, je nachdem, welche Taste man drückt. Das Einlesen der Variablen A$ verhindert einen SYNTAX ERROR bei Betätigung einer Buchstabentaste.

POKE36876,...: Die Tonhöhe wird gesetzt, und ein Ton wird ausgegeben, wenn A>0. Die Ausdrücke (A>1), (A>3), und so weiter bewirken eine Anpassung des Tonregisters an die Tonleiter, da sie bei erfüllter Bedingung den Wert —1 ergeben und im anderen Fall 0 sind.

GOTO1: Das Programm wird so lange wiederholt, bis die RUN/STOP-Taste gedrückt wird.

(Joachim Günther)

Track-Zerstörer: Kopierschutz (C 64/VC 20)

```
1 open1,8,15:open2,8,2,"#":print#1,"u1 2
 0";t;0:print#1,"m-e"chr$(163)chr$(253)
```

Diese Zeile produziert auf dem gewünschten Track der Diskette (Variable T) einen READ ERROR 21. Das bedeutet einen relativ sicheren Kopierschutz. (Jörg Wegmeyer)

Formatierte Ausgabe (C 64/VC 20)

```
20 b=abs(a):printtab(int(log(b-(b=0))*.4
3429448188)*(b>=1)+int(-b)*(b<1)+22);a
```

Dieser Einzeiler gibt Zahlen, egal welcher Größe und unabhängig vom Vorzeichen, rechtsbündig aus.

Die Zahl 22 gibt die Spalte minus 2 für die Kommastelle an. Wollen Sie zum Beispiel die Kommata in der Spalte 30 haben, so müssen sie statt 22 dann 28 (30-2) eingeben. A ist die Zahl die Sie ausgeben wollen.

Funktionsweise der Programmzeile:

Die Berechnung der Tabulatorfunktion unterteilt sich in die Bereiche für A größer-gleich eins, und A kleiner eins. Wobei A die Zahl ist, die ausgegeben werden soll.

Für A größer-gleich eins wird von A der Zehnerlogarithmus berechnet und von der Spalte abgezogen, bei der der Dezimalpunkt stehen soll. Da der Commodore 64 nur den Logarithmus zur Basis e berechnen kann, muß das Ergebnis mit der Konstante .43429448188 multipliziert werden. Dadurch erhält man den Zehnerlogarithmus.

Für diesen Teil der Berechnung ist nur der folgende Teil der Zeile maßgebend: INT(LOG(B-(B=0))*. 43429448188)*(B>=1)

Für Zahlen kleiner eins wird zu der Spalte eins dazuaddiert. Für Null entsteht ein Sonderfall, da die Null noch vor dem Komma stehen muß. Für diesen Teil ist der folgende Ausdruck maßgebend: INT(—B)*(B<1) (Volker Walter)

Programmierwettbewerb:

Das »intelligente« Programm

Computerprogramme tun sich ja in der Regel recht schwer, in normaler Umgangssprache eingegebene Sätze zu »verstehen« und womöglich noch folgerichtig zu reagieren. Viele Adventure-Programme zeigen aber zumindest ansatzweise, daß es doch geht.

Warum fällt dem Computer so schwer, was uns Menschen so leichtfällt, nämlich sich einfach zu unterhalten? Diese Frage ist gleichzeitig die Frage nach dem Wesen der Intelligenz. Eine Forschungsrichtung innerhalb der Informatik beschäftigt sich schon seit Jahren mit der Entwicklung und Erforschung von »Künstlicher Intelligenz«. Wir wollen Sie nun einmal dazu anregen, Programme zu entwickeln, die intelligentes Verhalten in einem gewissen Rahmen simulieren.

Die Eliza-Story

Im Jahre 1966 entwickelte Joseph Weizenbaum vom Massachusetts Institute of Technology ein Programm namens »Eliza«, das — vereinfacht gesagt — einen Psychoanalytiker simuliert. Der Mensch begibt sich also in der Rolle als Patient an die Tastatur des Eliza-Computers und wird aufgefordert, von seinen Schwierigkeiten zu berichten. Aufgrund der Eingaben gibt Eliza dann durchaus differenzierte Antworten oder stellt auch schon mal eine Zwischenfrage, so daß ein regelrechtes Zwiegespräch zustandekommt (siehe Bild).

Eliza wurde ursprünglich, wie fast alle Programme zur künstlichen Intelligenz, in der Programmiersprache Lisp geschrieben. Eine Basic-Version von Steve North wurde in den Siebziger Jahren in Creative Computing veröffentlicht. Das Bemerkenswerte an dem Eliza-Programm ist seine Einfachheit. Eliza wurde auch nicht als »intelligentes Programm« entwickelt, sondern sollte lediglich bewußt einfach menschliches (Gesprächs-) Verhalten simulieren. Die Eingabe des menschlichen Gesprächspartners wird nach bestimmten Schlüsselwörtern durchsucht, und zu jedem Schlüsselwort ist eine Reihe von möglichen Antworten vorgesehen.

Eliza wurde sehr schnell dadurch bekannt, daß ein großer Teil der Versuchspersonen der festen Überzeugung war, von einem anderen Terminal aus würde tatsächlich ein Psychiater die Antworten geben und dem Programm ihre ganz persönlichen Probleme anvertrauten. Aus diesen Versuchen wurde schließlich die Erkenntnis gezogen, daß die einfache Simulation menschlichen Verhaltens kein ausreichendes Kriterium für Intelligenz ist (so bitter das dem einen oder anderen erscheinen mag), denn Eliza analysiert ja in Wirklichkeit nicht die Satzstruktur, sondern akzeptiert auch völligen Unsinn.

Dennoch bleibt unbestreitbar eine gewisse Faszination, wenn man einmal mit Eliza gearbeitet hat. Diese Faszination wollen wir auch ins 64'er Magazin holen. Setzen Sie sich doch einmal an Ihren Schreibtisch und überlegen Sie sich, wie ein Programm funktionieren müßte, mit dem man sich in ganzen Sätzen unterhalten kann.

Um die ganze Angelegenheit nicht zu kompliziert zu machen, braucht das Programm nur einfache deutsche Sätze bearbeiten, das heißt es muß bei Eingabe von Goethes »Faust« nicht unbedingt von sich aus den »Faust II« zitieren. Außerdem können Sie den Themenbereich, in dem das Programm sicher arbeitet, durchaus auf bestimmte Gebiete einschränken. Beispielsweise wäre ein Programm denkbar, das mit dem Menschen über die Vor- und Nachteile von Computern diskutiert. Seien Sie bei der Themenwahl (und beim Programmieren) ruhig kreativ.

Das beste Programm, das sich mit uns Redakteuren unterhalten will, ist uns 1000 Mark wert. Länge und Programmierstil sind unwichtig, entscheidend ist der Effekt.

Senden Sie uns Ihr Programm für den C 64 oder VC 20 auf Diskette oder Kassette zu, und vergessen Sie nicht, eine ausreichende Beschreibung beizufügen. Unsere Adresse ist:

Redaktion 64'er, Stichwort »Programmierwettbewerb: Das »intelligente« Programm«, Markt & Technik Verlag AG, Hans-Pinsel-Str. 2, 8013 Haar bei München.

Der Beginn einer Sitzung mit »Eliza«

```
HI, I AM ELIZA
PLEASE TELL ME YOUR PROBLEM
I HATE COMPUTERS
DO COMPUTERS WORRY YOU?
YES
ARE YOU SURE?
I DON'T REALLY KNOW
DON'T YOU REALLY REALLY KNOW
HA HA
SAY, DO YOU HAVE ANY
PSYCHOLOGICAL PROBLEMS?
ELIZA YOU ARE MY PROBLEM
WHAT MAKES YOU THINK I AM
YOUR PROBLEM
YOUR SYNTAX ERRORS
WHY ARE YOU CONCERNED ABOUT
MY SYNTAX ERRORS
I HATE THEM
WHAT DOES THAT SUGGEST TO YOU?
I DON'T KNOW
WHY DON'T YOU KNOW
I CAN'T THINK ABOUT IT
HOW DO YOU KNOW YOU CAN'T
THINK ABOUT IT
YOU ASK FUNNY QUESTIONS
WE WERE DISCUSSING YOU — NOT ME!
```

Fortsetzung von Seite 22.

Ein starkes Stück

gramm (Vizawrite 64). Außer einem Verbindungskabel für knapp 50 Mark waren keine zusätzlichen Maßnahmen notwendig. Der Ausdruck funktionierte auf Anhieb mit der eingebauten Treibersoftware des Textverarbeitungsprogramms. Die zweite Frage galt der Grafikfähigkeit der Itoh Drucker. Aber auch hier hatten »Die Zwei« eine Überraschung parat. Je nach der eingestellten Schriftart (Pitchsetting) sind Punktdichten von 1088 bis zu 2176 Punkten pro Zeile beim 1550B möglich. Diese Auflösung wird vom Commodore natürlich bei weitem nicht erreicht. Die Deutlichkeit der Grafik steigt aber mit zunehmender Punktdichte und es sind auch mehrere Farbschattierungen (Graustufen) als Farbersatz programmierbar. Die zweite Methode, andere Zeichen als die vom Zeichengenerator vorgegebenen auszudrucken, ist der ladbare Zeichensatz. Die eingebauten 2-KByte-Pufferspeicher sind auch dazu verwendbar, einen eigenen Zeichensatz zu programmieren. In Verbindung mit dem ebenfalls möglichen Rückwärtstransport des Papiers, kann ähnlich wie mit einem Plotter gearbeitet werden.

Alles in allem stellen die beiden Itoh Drucker 1550B und 8510 ein sehr leistungsfähiges Druckerpaar dar. Die verfügbaren Schriften haben ein überdurchschnittlich gutes Druckbild. Einzig eine gedehnte Schrift, wie von den Commodore- und Epson-Druckern bekannt, fehlt. Bei einem Preis von 2400 Mark für den 1550B und 1800 Mark für den 8510 stellen die Itoh-Drucker eine kostengünstige Alternative dar.

(Arnd Wängler)



Floppy-Kurs

Fortsetzung von Seite 120

Auch dieser Befehl ist äquivalent zu einem Basic-Befehl, dem SYS-Befehl. Mit ihm kann man also ein Maschinenprogramm an einer beliebigen Stelle im Floppy-Speicher ausführen. Syntax:
PRINT #fn,"M-E"CHR$(adl)CHR$ (adh).
Siehe auch Listing 7.

Die USER-Befehle

Die USER-Befehle stellen eine Erweiterung des Befehlssatzes dar, der fast ausschließlich der Bequemlichkeit dient. U1 und U2 wurden schon besprochen, sie ersetzen B-R und B-W. Die Befehle U3 bis U8 dienen dem Start eines Maschinenprogramms im Floppy-Speicher, dessen Anfangsadressen in einer Tabelle abgelegt sind, so entsprechen:

U3 einem Start bei $0500
U4 einem Start bei $0503
U8 einem Start bei $050F.

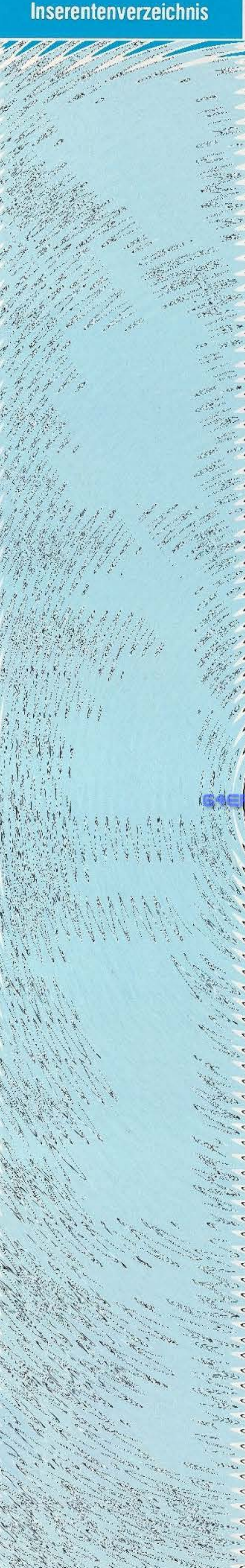
U4 ersetzt also beispielsweise den Befehlsstring: M-E CHR$(3)CHR$(5). U9 zeigt auf den NMI-Vektor der 1541, welcher allerdings eine Sonderfunktion hat: Mit U9+ wird die Floppy auf C 64- und mit U9- auf VC 20-Betrieb umgeschaltet.

U: stellt einen Reset dar, ähnlich dem SYS 64738 beim C 64.

Mit den Kenntnissen über den Befehlssatz der VC 1541 dürfte es Ihnen nun keine Schwierigkeiten mehr bereiten, sich das Programm EDDI einmal zu Gemüte zu führen. Das einzig Besondere daran sind die Routinen zum Lesen und Schreiben eines Blocks, die aus Geschwindigkeitsgründen in Maschinensprache geschrieben sind. Ein großer Teil der in diesen Folgen erwähnten Informationen ist auch im Commodore-Handbuch enthalten, nur sind dort oft Fehler.

Wir wollen uns für diese Folge von Ihnen verabschieden, nicht ohne Sie dringend dazu anzuhalten zu probieren. (Schramm/Schneider/gk)

Inserentenverzeichnis



Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Michael M. Pauly (py)
Stellv. Chefredakteur: Michael Scharfenberger (sc)
Redakteure: aa = Albert Absmeier, leitender Redakteur, ev = Volker Everts, gk = Georg Klinge, rg = Christian Rogge
Redaktionsassistenz: Gerda Sigl (202)
Fotografie: Janos Feitser, Titelfoto: Alex Kempkens

Layout: Leo Eder (Ltg.), Dagmar Berninger, Willi Gründl, Cornelia Weber

Auslandsrepräsentation:
Schweiz: Markt&Technik Vertriebs AG, Alpenstrasse 14, CH-6300 Zug, Tel. 042-2231 55/56, Telex: 862329 mut ch
USA: M&T Publishing, 2464 Embarcadero Way, Palo Alto, CA 94303; Tel. 001-4240600; Telex 752351

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten werden, so muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlags AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Herstellung: Klaus Buck (180), Leo Eder (181)

Anzeigenleitung: Peter Schrödel (156)
Anzeigenverkauf: Alfred Reeb (211)

Anzeigenverwaltung und Disposition: Michaela Hörl (171)

Anzeigenformate: ¹/₁-Seite ist 266 Millimeter hoch und 185 Millimeter breit (3 Spalten à 58 mm oder 4 Spalten à 43 Millimeter). Vollformat 297x210 Millimeter. Beilagen und Beihefter siehe Anzeigenpreisliste.

Anzeigenpreise: Es gilt die Anzeigenpreisliste Nr. 1 vom 1. März 1984.
Anzeigengrundpreise: ¹/₁ Seite sw: DM 7400,-. Farbzuschlag: erste und zweite Zusatzfarbe aus Europaskala je DM 1000,-. Vierfarbzuschlag DM 3000,-. Plazierung innerhalb der redaktionellen Beiträge: Mindestgröße ¹/₄-Seite
Anzeigen im Einkaufs-Magazin: Die ermäßigten Preise im Einkaufs-Magazin gelten nur innerhalb des geschlossenen Anzeigenteils, der ohne redaktionelle Beiträge ist. ¹/₁-Seite sw: DM 5400,-. Farbzuschlag: erste und zweite Zusatzfarbe aus Europaskala je DM 1000,-. Vierfarbzuschlag DM 3000,-. **Anzeigen in der Fundgrube: Private Kleinanzeigen** mit maximal 5 Zeilen Text DM 5,- je Anzeige. **Gewerbliche Kleinanzeigen:** DM 10,- je Zeile Text.
Auf alle Anzeigenpreise wird die gesetzliche MwSt. jeweils zugerechnet.

Vertriebsleitung, Werbung: Hans Hörl (114)

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Plieninger Straße 100, 7000 Stuttgart 80 (Möhringen), Telefon (07 11) 72004-0

Erscheinungsweise: 64'er, Magazin für Computerfans, erscheint monatlich, Mitte des Vormonats.

Bezugsmöglichkeiten: Leser-Service: Telefon 089/46 13-1 19. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen. Das Abonnement verlängert sich zu den dann jeweils gültigen Bedingungen um ein Jahr, wenn es nicht zwei Monate vor Ablauf schriftlich gekündigt wird.

Bezugspreise: Das Einzelheft kostet DM 6,-. Der Abonnementspreis beträgt im Inland DM 72,- pro Jahr für 12 Ausgaben. Darin enthalten sind die gesetzliche Mehrwertsteuer und die Zustellgebühren. Der Abonnementspreis erhöht sich um DM 18,- für die Zustellung im Ausland, für die Luftpostzustellung in Ländergruppe 1 (z.B. USA) um DM 38,-, in Ländergruppe 2 (z.B. Hongkong) um DM 58,-, in Ländergruppe 3 (z.B. Australien) um DM 68,-.

Druck: Druckerei E. Schwend GmbH, Schmollerstr. 31, 7170 Schwäbisch Hall

Verantwortlich: Für redaktionellen Teil: Michael M. Pauly.
Für Anzeigen: Peter Schrödel.

Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/46 13-0, Telex 522052

Telefon-Durchwahl im Verlag:

Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089-4613 und dann die Nummer, die in Klammern hinter dem jeweiligen Namen angegeben ist.

VORSCHAU

Ein neuer Kurs über Musik

Der SID vom C 64 wird oft als der leistungsfähigste Synthesizer-Chip in einem Heimcomputer bezeichnet; wir meinen mit recht. Was liegt also näher als mit einem Kurs zu beginnen, der Ihnen diesen elektronischen Musikus aufschlüsselt.

Test: Lernsoftware

Sie ist im Kommen, die Software, mit der man mühelos mittels des C 64 Vokabeln pauken, Fremdsprachen und Basic erlernen oder mathematische Probleme lösen kann. Also weg vom Spielen und hin zum spielenden Lernen.

Der mathematische VC 20

»Mathematical Basic«, eine umfangreiche Basic-Erweiterung für den VC 20 mit über 40 neuen Befehlen und Funktionen wurde diesmal Listing des Monats. Neben zusätzlichen mathematischen Funktionen gibt es eine Reihe von Befehlen zum einfacheren Arbeiten mit der Diskettenstation und weitere Programmierhilfen.

Das Spiel des Jahres

Es gab 1984 eine wahre Flut an neuen Spielen für den C 64. Ein Spiel hat uns jedoch 1984 besonders beeindruckt, weil es nicht nur für kurze Zeit interessant ist, sondern weil man es um so lieber spielt, je länger man es hat. Warum? — Lassen Sie sich überraschen.

Synthesizer

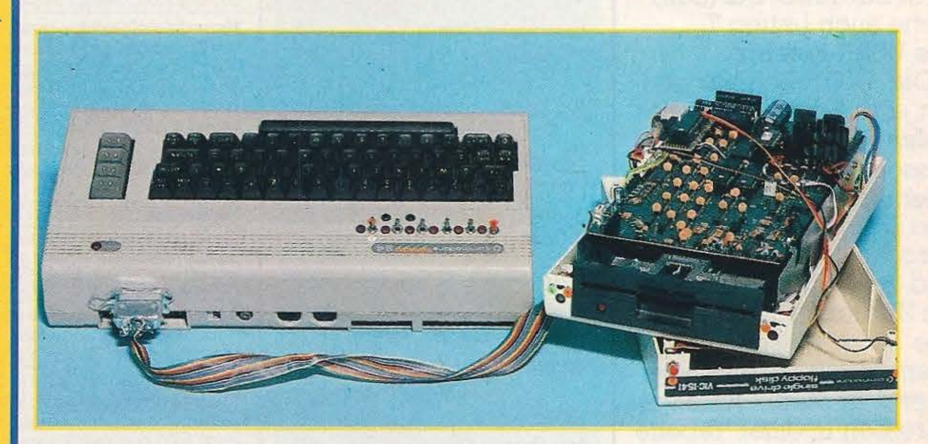
Unser Synthesizer zum Abtippen ist kein einfaches Orgelprogramm. Mit ihm lassen sich alle möglichen Einstellungen des Sound-Chips simulieren. Sie können sogar dreistimmig spielen. Ein Programm, das keine Wünsche offen läßt.

Print 64 — ein außergewöhnliches Interface

Centronics-Schnittstellen für den Commodore 64 gibt es einige auf dem Markt. Auch Print 64 ist sicherlich noch nicht der Weisheit letzter Schluß. Dennoch überrascht es mit einigen erfreulichen Leistungsmerkmalen.

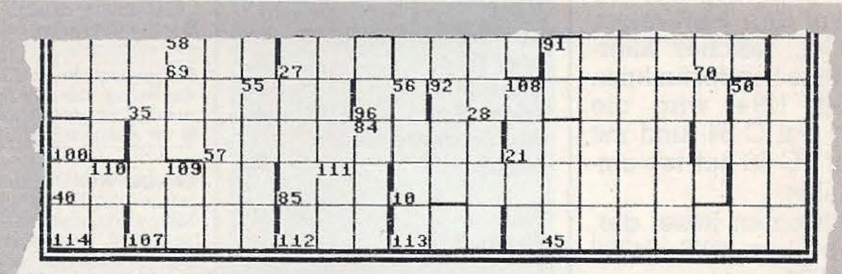
C 64 und Floppy schneller gemacht

Während das HYPRA-LOAD in Ausgabe 10/84 eine reine Software-Lösung war, werden jetzt Systeme vorgestellt, die zusätzlich auch Hardware benötigen. Dabei beschränkt man sich nicht nur darauf, die schnellermachende Software auf EPROMs zu brennen, sondern die Daten werden nun parallel transportiert. Doch wie ist es mit der Kompatibilität bestellt? Ist der Drucker ansprechbar? Lohnt sich also eine Umrüstung?



Kreuzworträtsel-Wettbewerb

Der Sieger war ziemlich schnell gefunden. Dessen Programm unterschied sich in einigen wesentlichen Merkmalen von den anderen. Es besitzt eine sehr komfortable Wörterauswahl, erstellt eine hervorragende Druckerausgabe und ist auch noch das schnellste Programm gewesen.



Vergleichstest Joysticks

Mit der Spiele-Welle kam auch die Joystick-Welle. Wir haben die verbreitetsten Joysticks für C 64 und VC 20 getestet. Dieser Test stellt eine wirkliche Hilfe bei der Auswahl eines Joysticks dar.




64ER ONLINE




ISSN ONLINE